

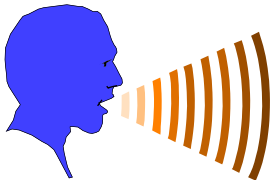



Image and Video Processing

Noise Removal, Edge Detection and Image Sharpening

Yao Wang
Polytechnic School of Engineering, New York University

Summary of Previous Lecture

- 2D CSFT and DSFT
  - Think of transform as representing a signal as weighted average of selected orthonormal basis functions
  - 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
  - Frequency response of the filter
- MATLAB function: conv2( ), freqz2( )

Outline

- Linear filtering for typical image processing applications
  - Noise removal
  - image sharpening
  - edge detection
- Median filtering
- Morphological filtering

Convolution Theorem

- Convolution Theorem

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

- 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

$$\begin{aligned}
G(u,v) &= \sum_{m,n} \sum_{k,l} f(m-k, n-l) h(k,l) e^{-j2\pi(mu+nv)} \\
&= \sum_{m,n} \sum_{k,l} f(m-k, n-l) e^{-j2\pi((m-k)u+(n-l)v)} h(k,l) e^{-j2\pi(ku+lv)} \\
&= \sum_{m,n} f(m-k, n-l) e^{-j2\pi((m-k)u+(n-l)v)} \sum_{k,l} h(k,l) e^{-j2\pi(ku+lv)} \\
&= \sum_{m',n'} f(m',n') e^{-j2\pi(m'u+n'v)} \sum_{k,l} h(k,l) e^{-j2\pi(ku+lv)} \\
&= F(u,v) \times H(u,v)
\end{aligned}$$

Another view of convolution theorem

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

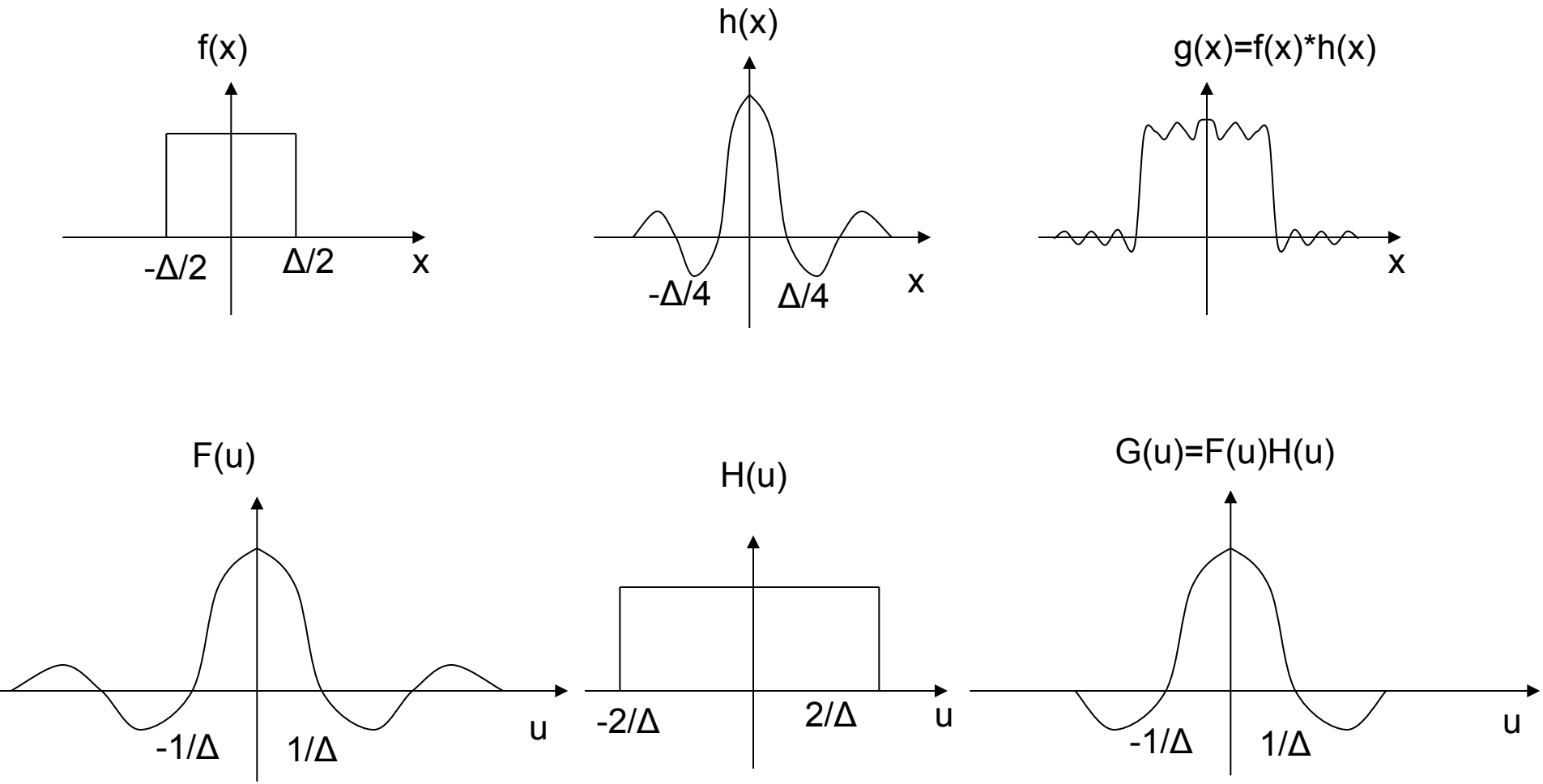
- F(u,v)H(u,v) = Modifying the signal's each frequency component' complex magnitude F(u,v) by H(u,v)

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

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

- H(u,v) is also called Frequency Response of the 2D LSI system
  - exp{2p (um+vn)} -> H(u,v) exp{2p (um+vn)} = |H(u,v)|exp{2p (um+vn)+P(H(u,v))}

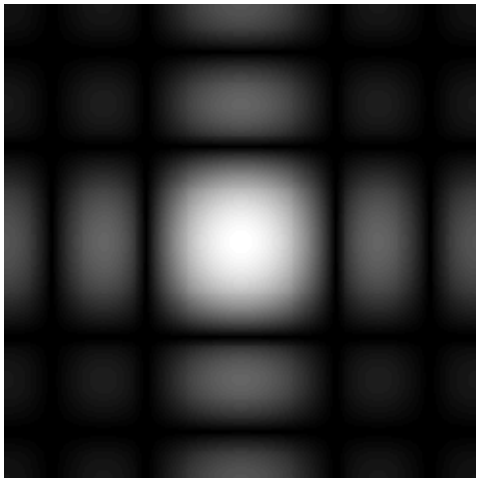
Explanation of Convolution in the Frequency Domain



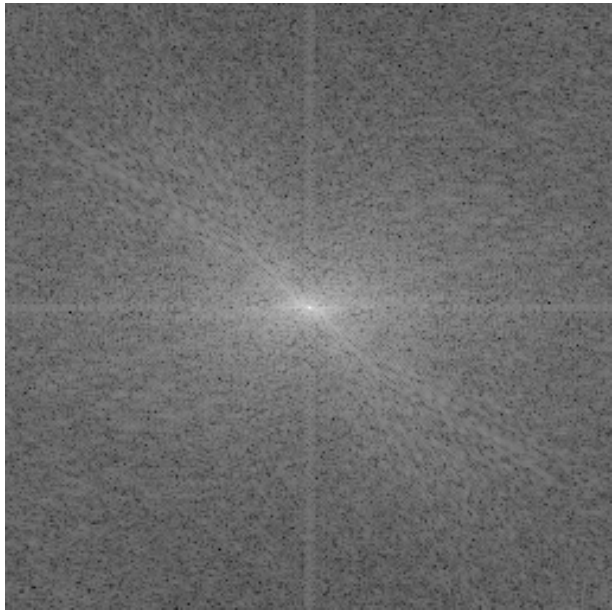
Example

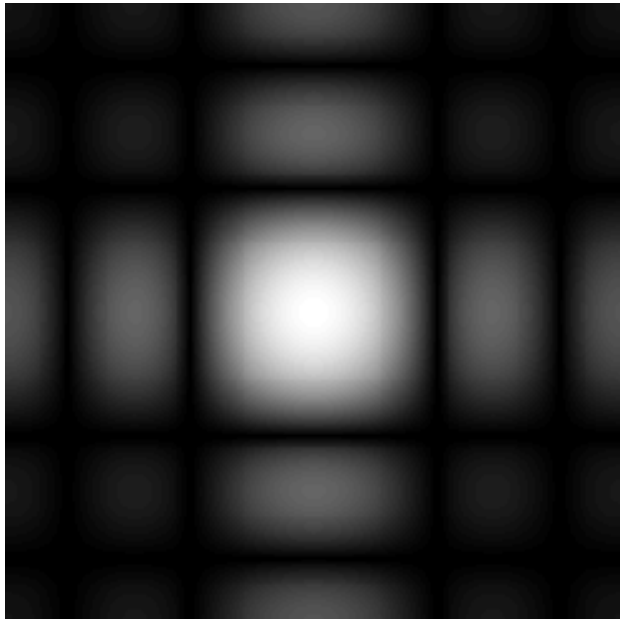
- Given a 2D filter, determine its frequency response. Apply to a given image, show original image and filtered image in pixel and freq. domain

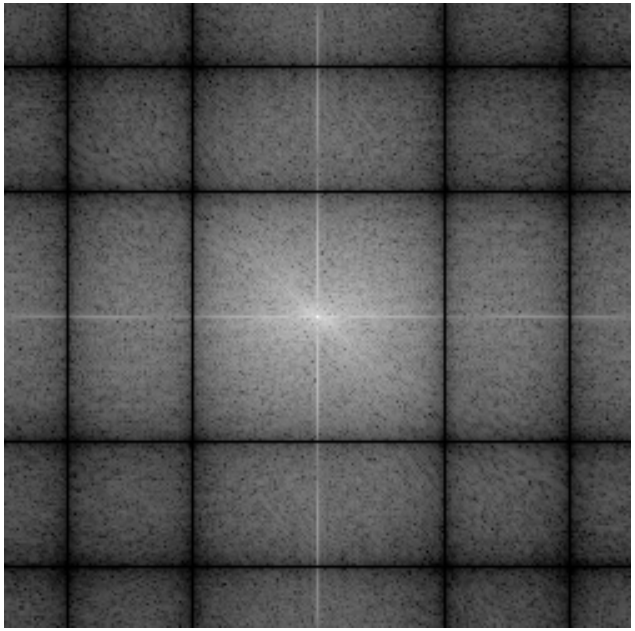
$$h = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



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



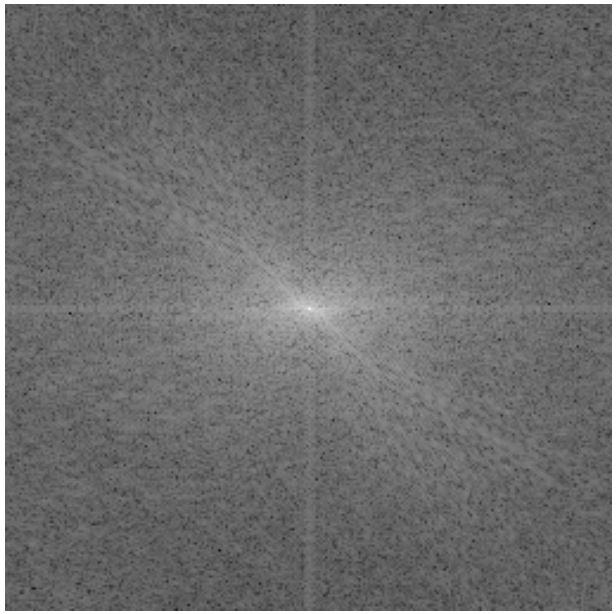


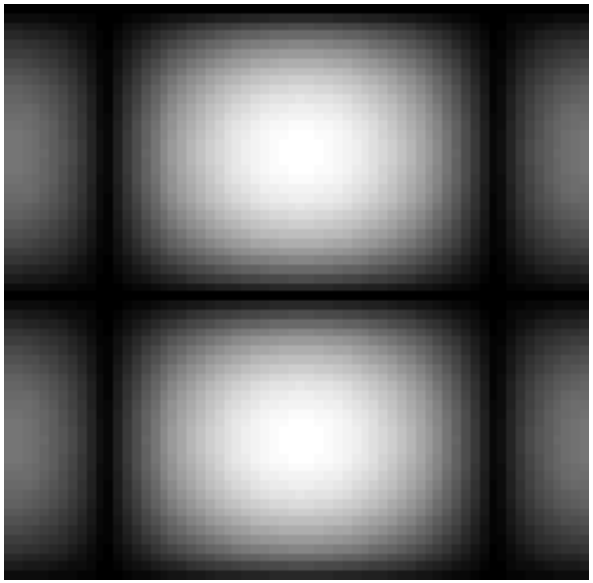


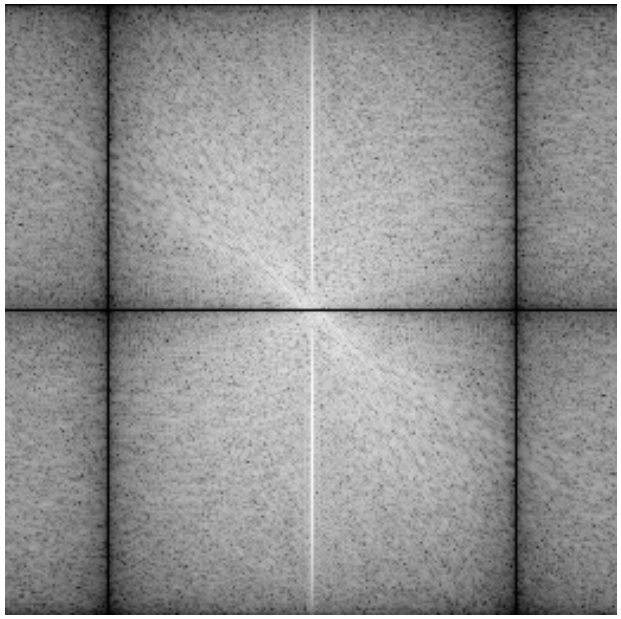
Matlab Program Used

```
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;
```

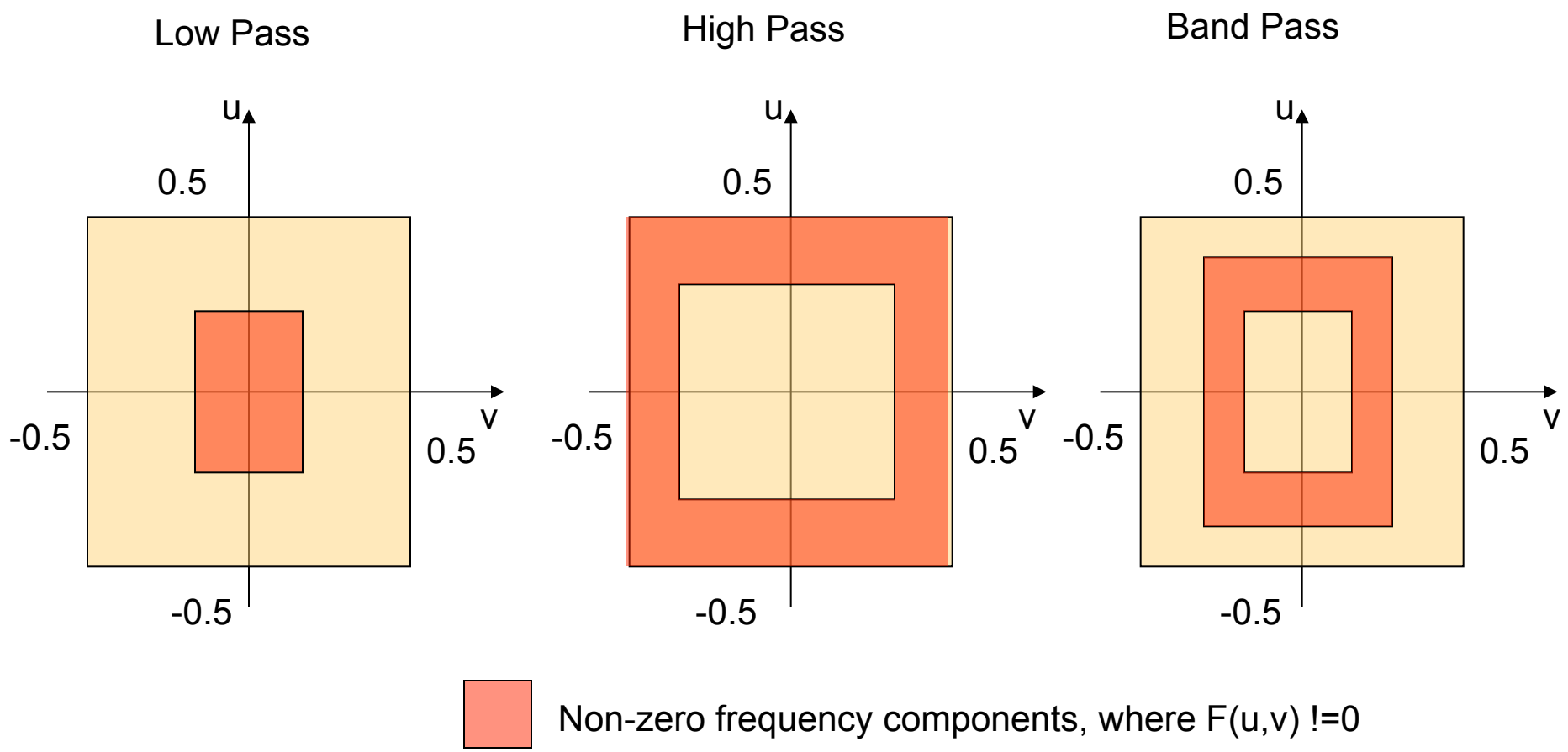
$$H_1 = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$







Typical Filter Types



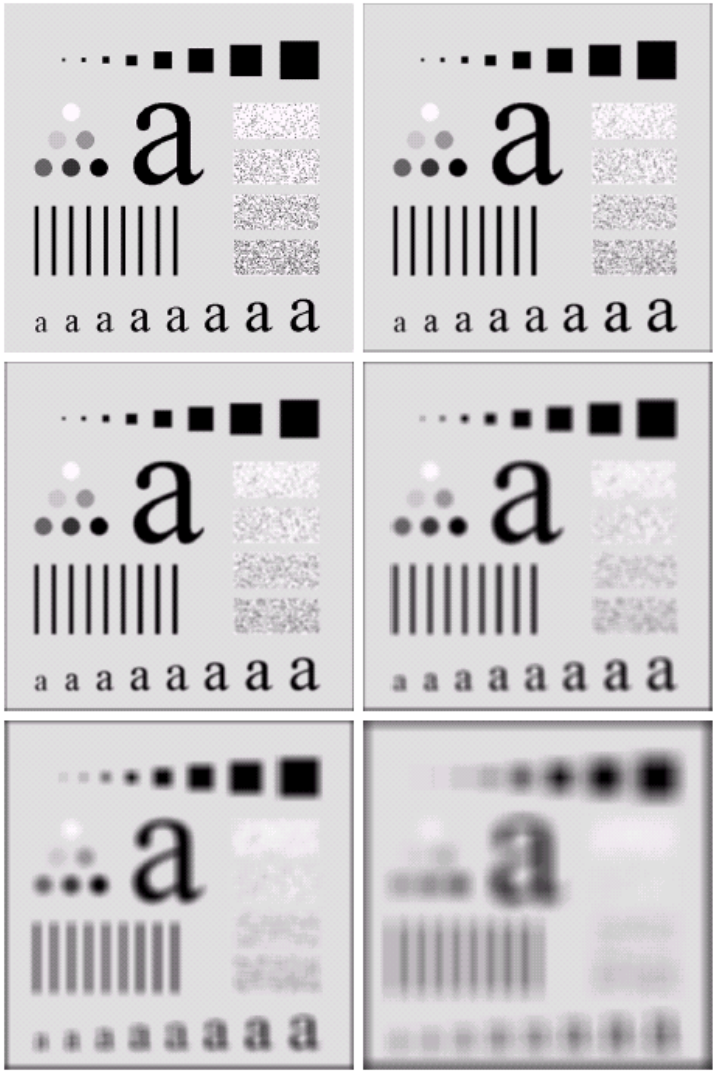
Filter Design

- 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
  - Inverse FT to get spatial filter
  - May further truncate
- Other filter design methods using optimization techniques

Typical Image Processing Tasks

- Noise removal (image smoothing): low pass filter
- Edge detection: high pass filter
- 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

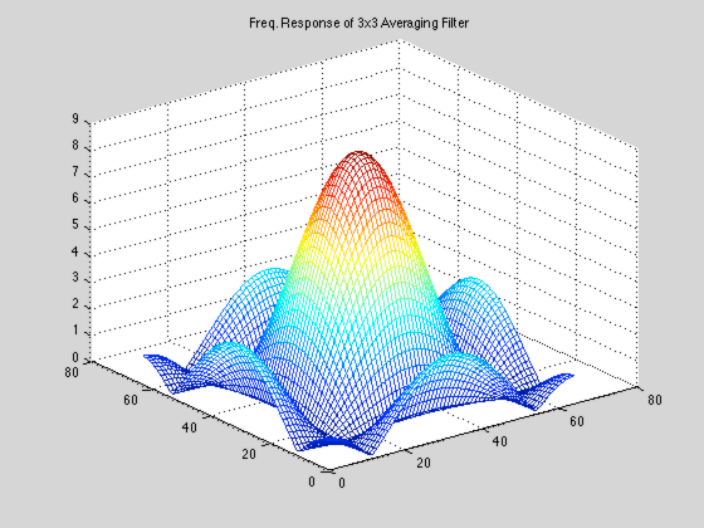


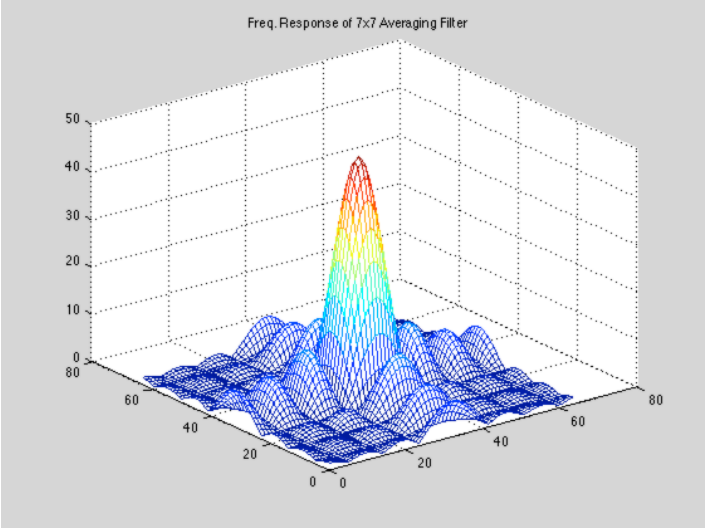
a b
c d
e f

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.

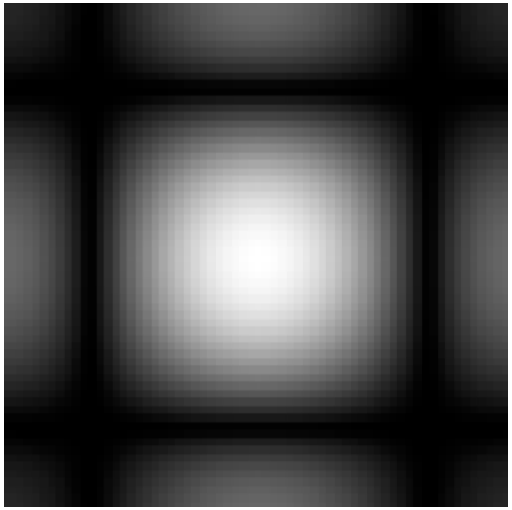
Window size controls tradeoff between noise removal power and blurring

Freq. Response Corresponding to Averaging Filters of Different Sizes

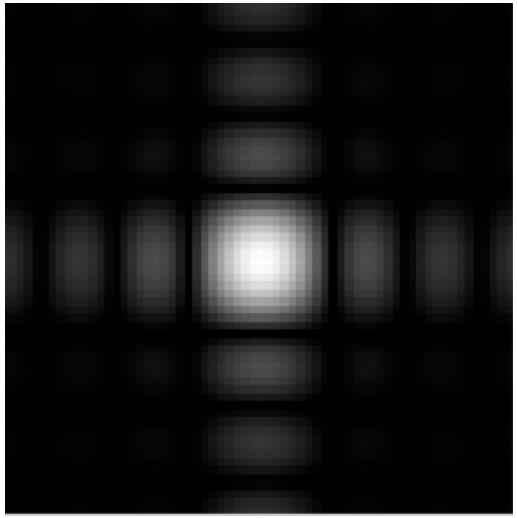




3x3 filter



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

- Analog form: STD σ controls the smoothing strength

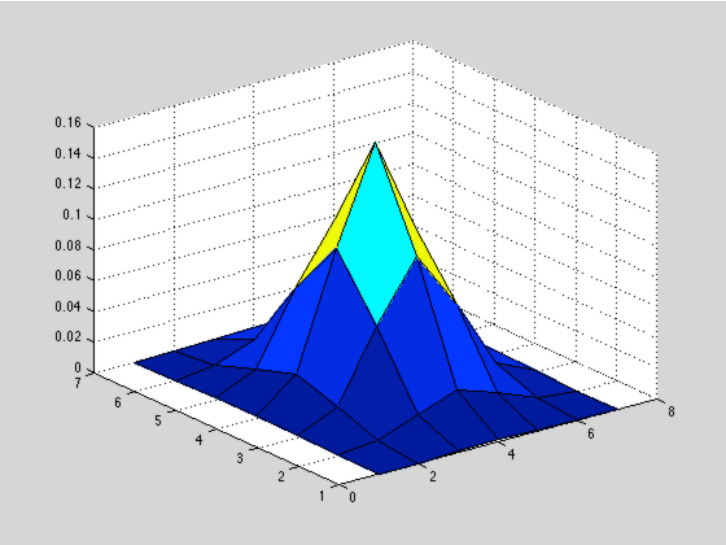
$$h(x,y) = \alpha \exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\},$$

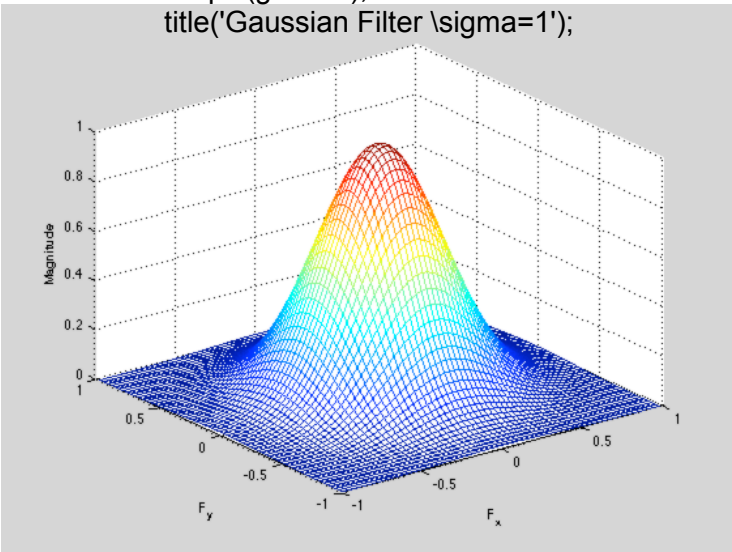
- Take samples, truncate after a few STD, normalize the sum to 1. Usually $\sigma >= 1$
- Size of mask nxn, n>=5s, odd
  - Ex: $\sigma=1, n=7$.
  - Show filter mask,
  - Show frequency response

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 Freq. Domain

- Still a Gaussian Function!
- 1D Gaussian filter

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

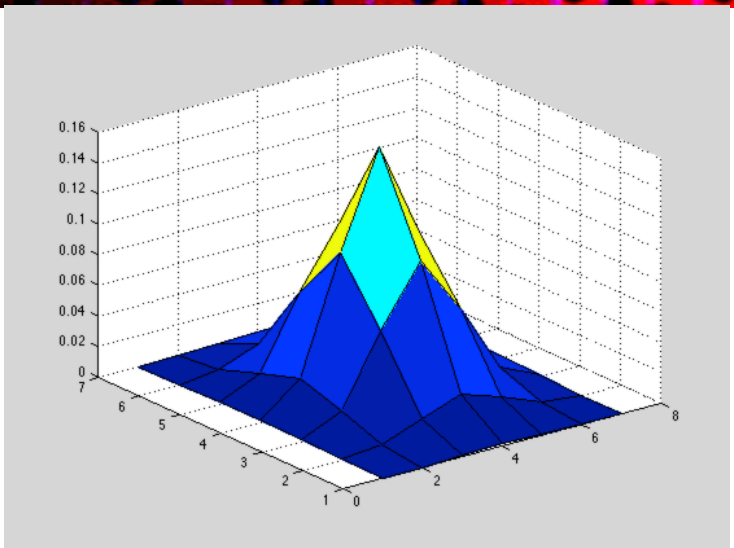
- 2D Gaussian filter

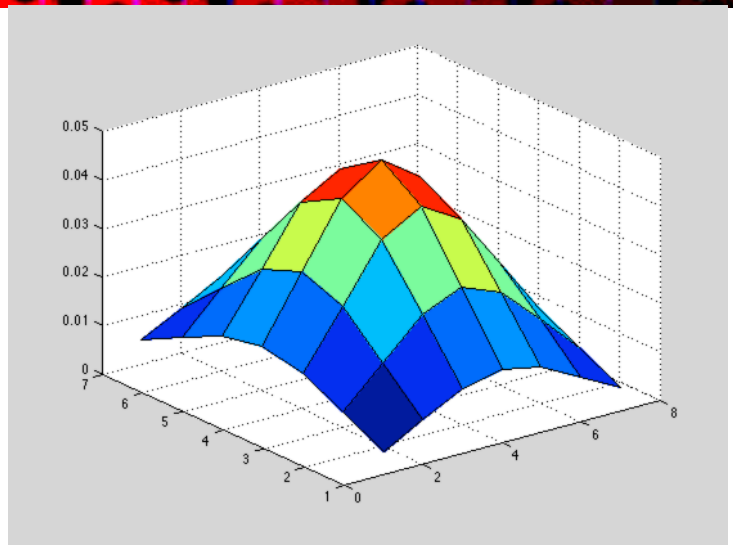
$$\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\} \Leftrightarrow \exp\left\{-\frac{u^2+v^2}{2\beta^2}\right\}, \beta = \frac{1}{2\pi\sigma}$$

- Note that STD b in freq. inversely related to STD a in space

Gaussian Filter in Space and Freq.

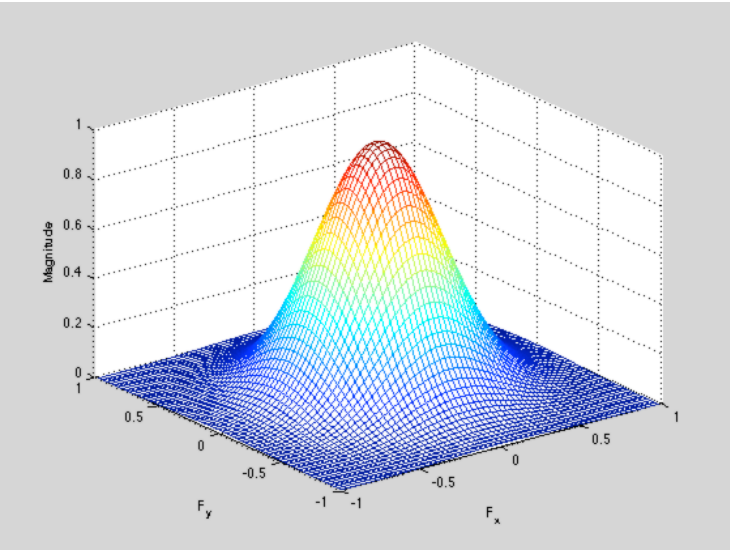
σ=1

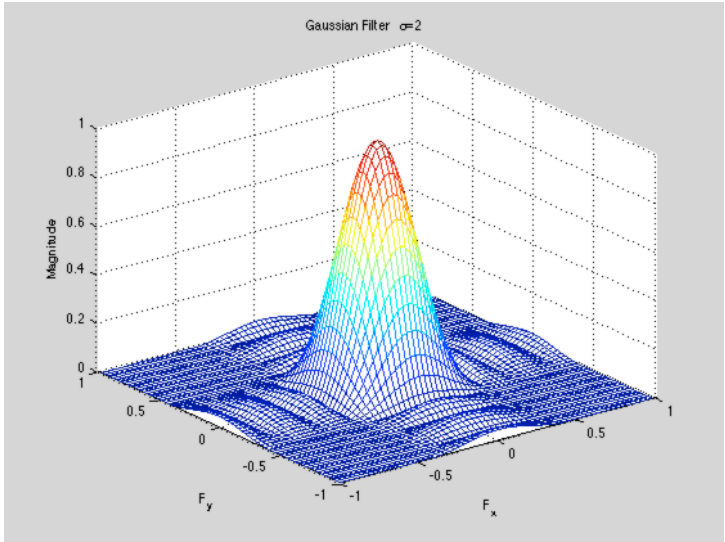




Σ=2

β=0.16

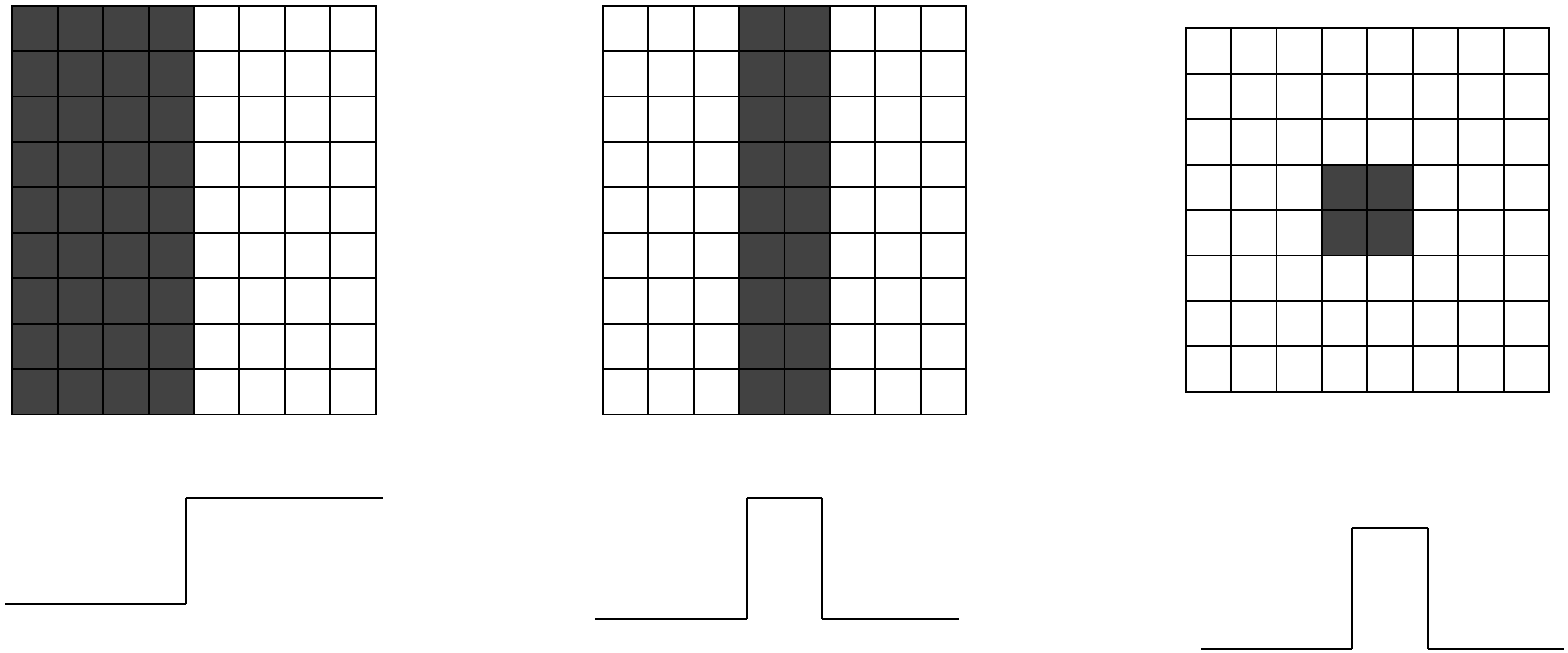




β=0.08

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 have have step edges on both sides



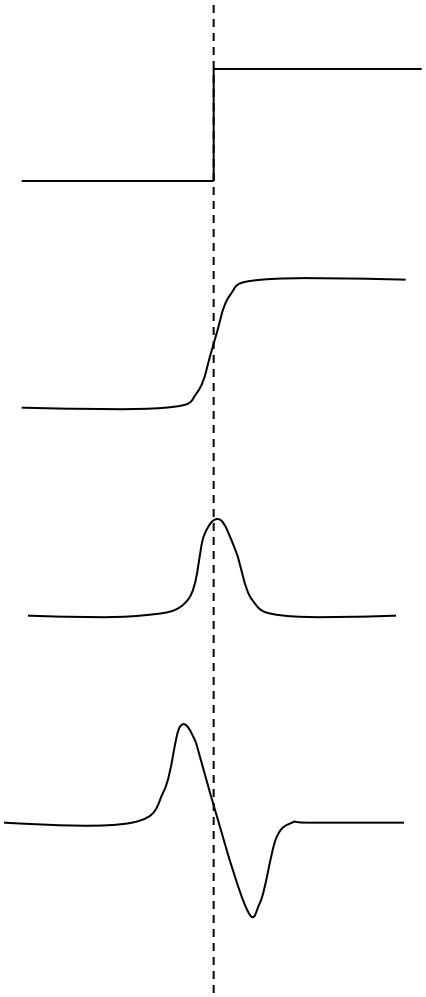
Characterization of Edges

Ideal step edge

Real edge has a slope

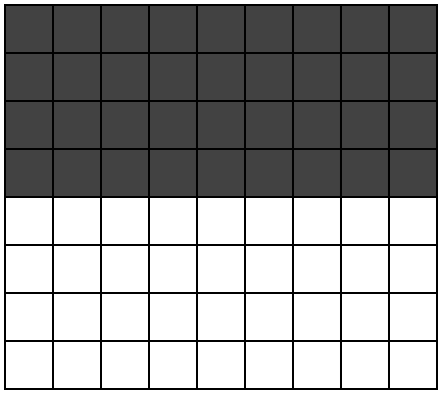
First order derivative:
Maximum at edge location

Second order derivative:
Zero crossing at edge location



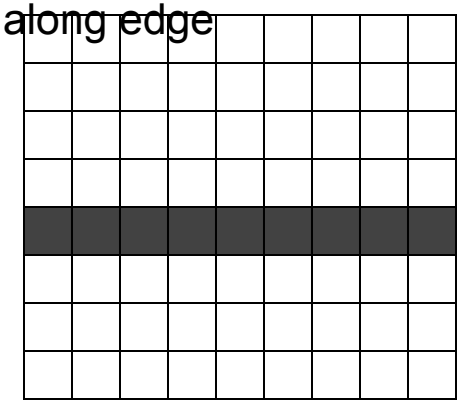
Edge Detection Based on First Order Derivatives

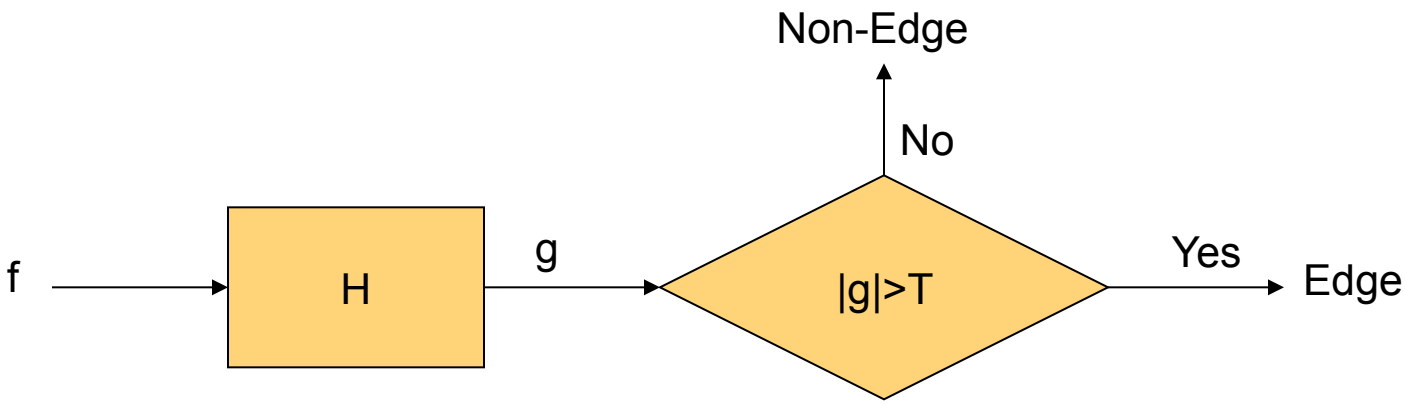
- Edge



High-pass filtering across edge
Low-pass filtering along edge

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

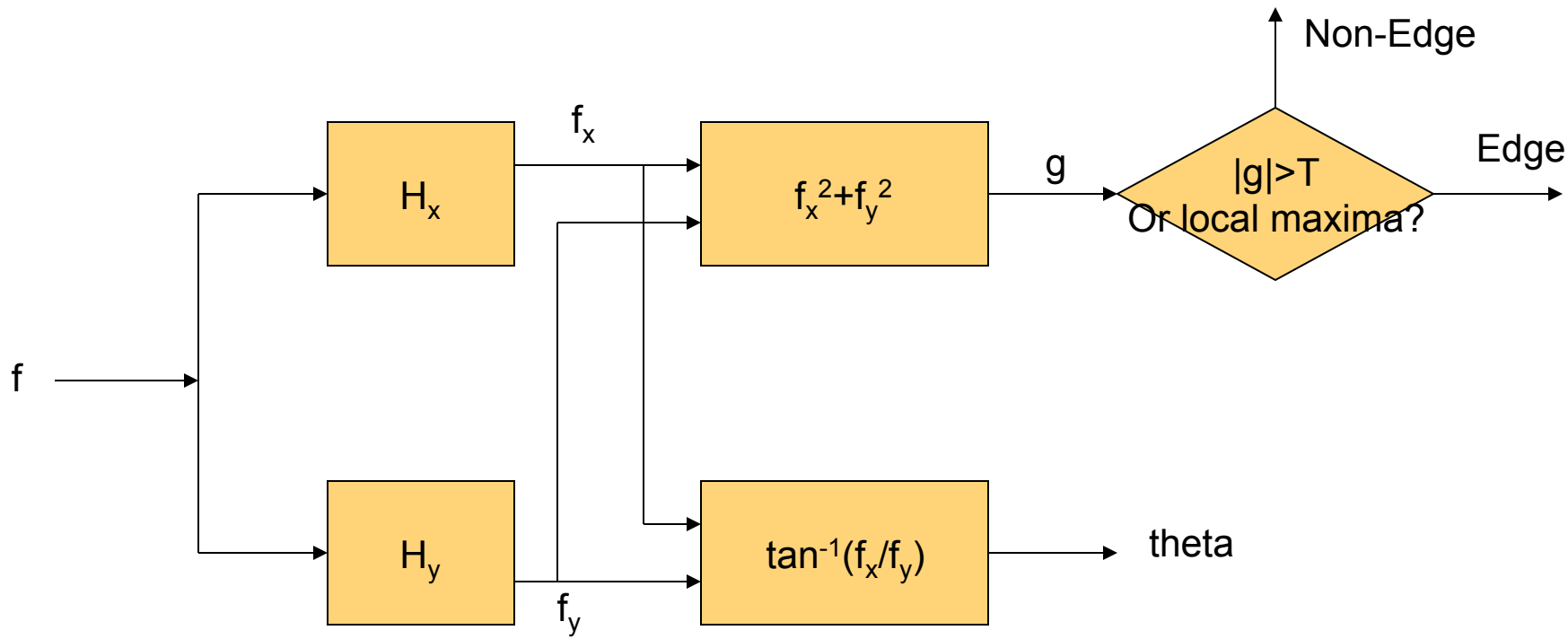




What if we don't know edge direction?

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

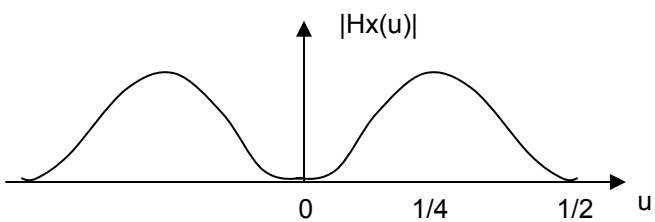
Sobel Filter for Horizontal Edges :

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

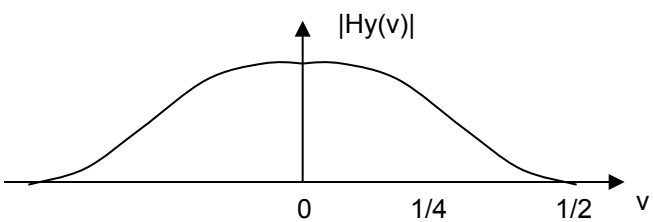
Frequency Response (DTFT) :

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

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



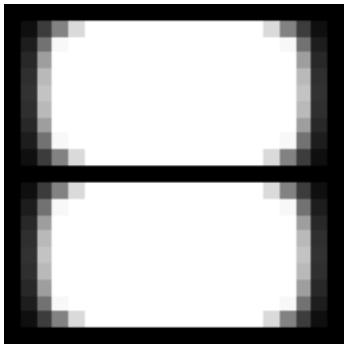
Band-pass

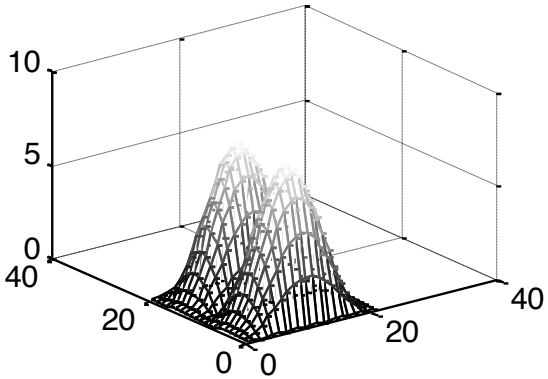


Low-pass

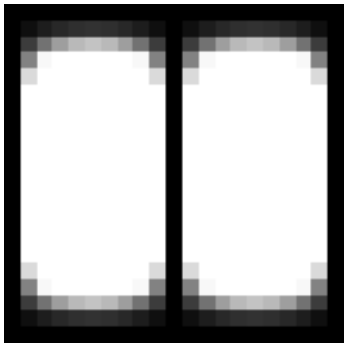
Spectrum of the Sobel Filter

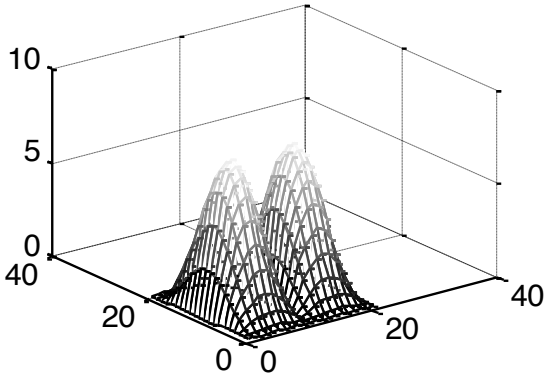
H_x





H_y





Low pass along the edge, band pass cross the edge

Example of Sobel Edge Detector

Original image

Filtered image by H_x

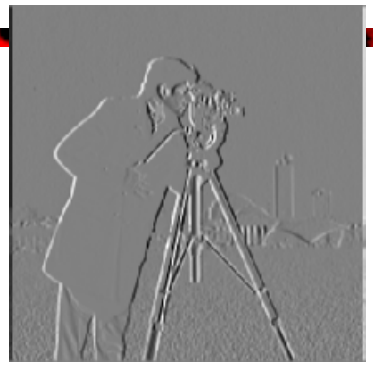
Filtered image by H_y

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

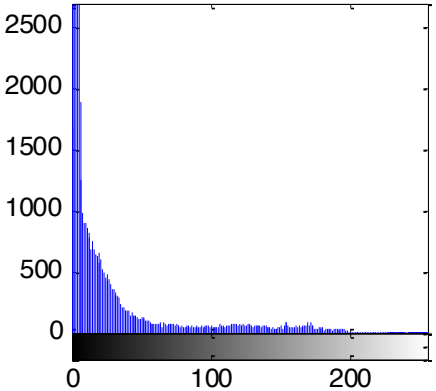
gx

gy

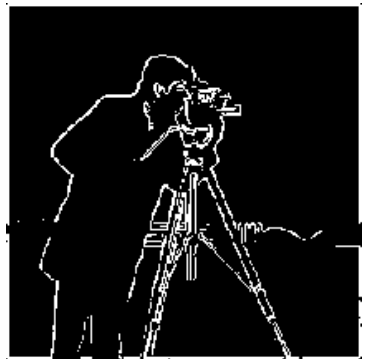


gm

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 Difference 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

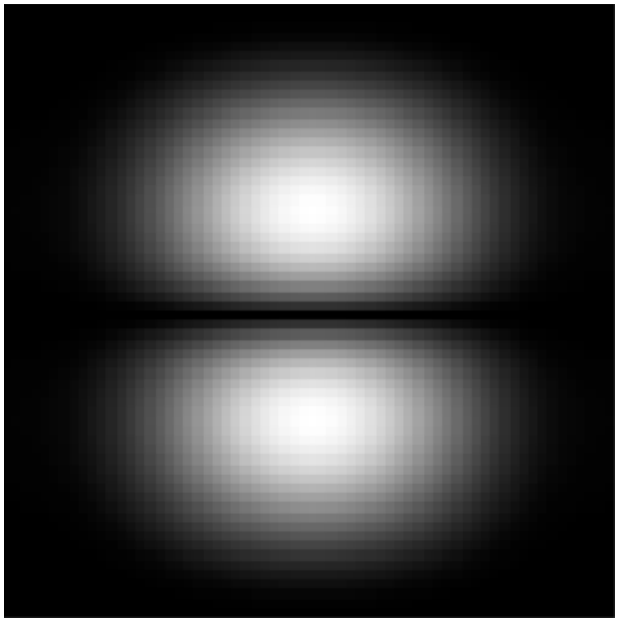
DoG Filter Examples

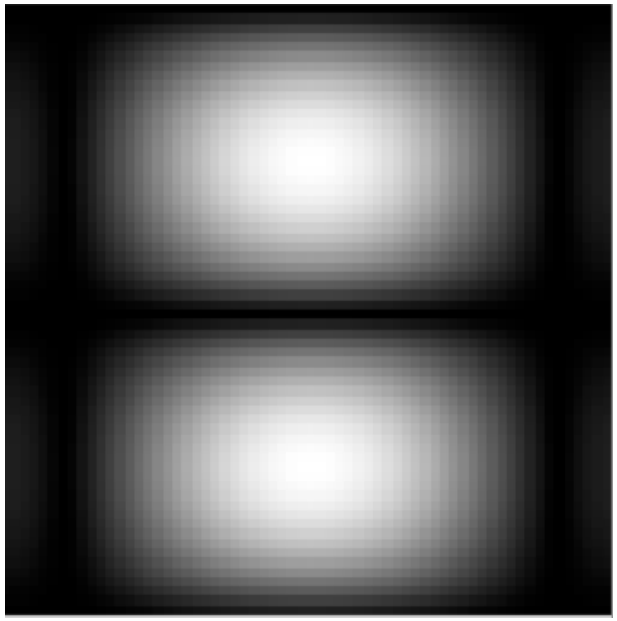
s=1, n=5

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 band-pass filters in one direction, low-pass in orthogonal direction

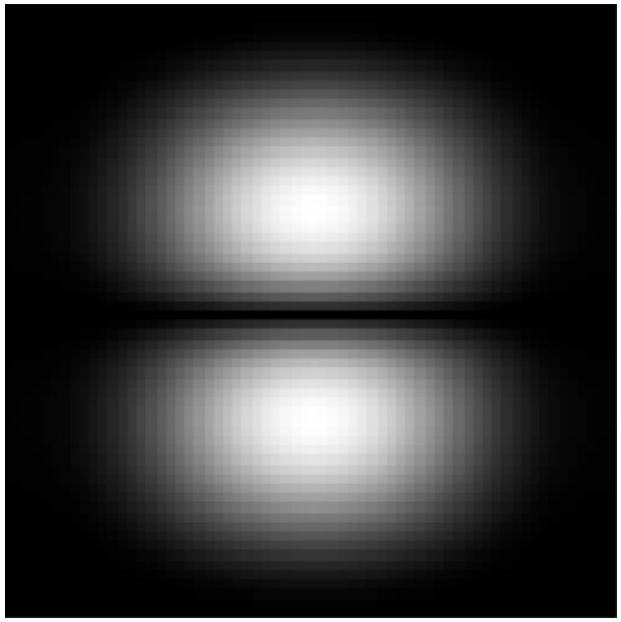
DoG Filter Examples

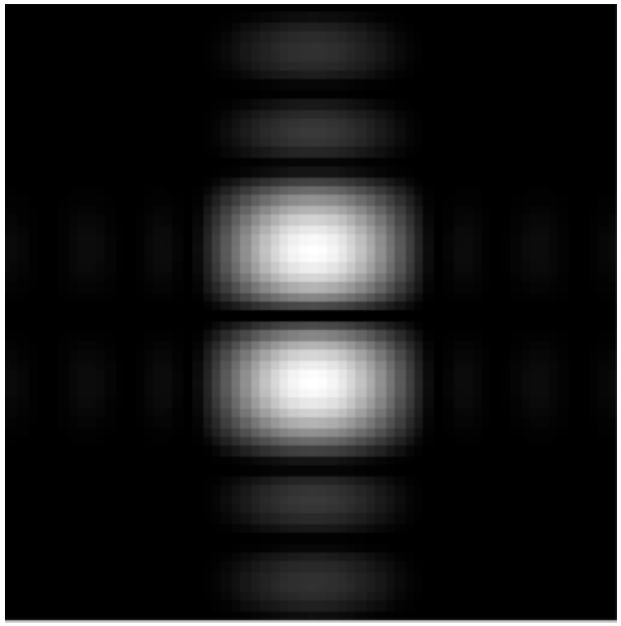
s=1, n=5

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=2, n=7 (more smoothing)

0.0790	0.1477	0.2149	0.2435	0.2149	0.1477	0.0790
0.0985	0.1839	0.2676	0.3033	0.2676	0.1839	0.0985
0.0716	0.1338	0.1947	0.2206	0.1947	0.1338	0.0716
0	0	0	0	0	0	0
-0.0716	-0.1338	-0.1947	-0.2206	-0.1947	-0.1338	-0.0716
-0.0985	-0.1839	-0.2676	-0.3033	-0.2676	-0.1839	-0.0985
-0.0790	-0.1477	-0.2149	-0.2435	-0.2149	-0.1477	-0.0790





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

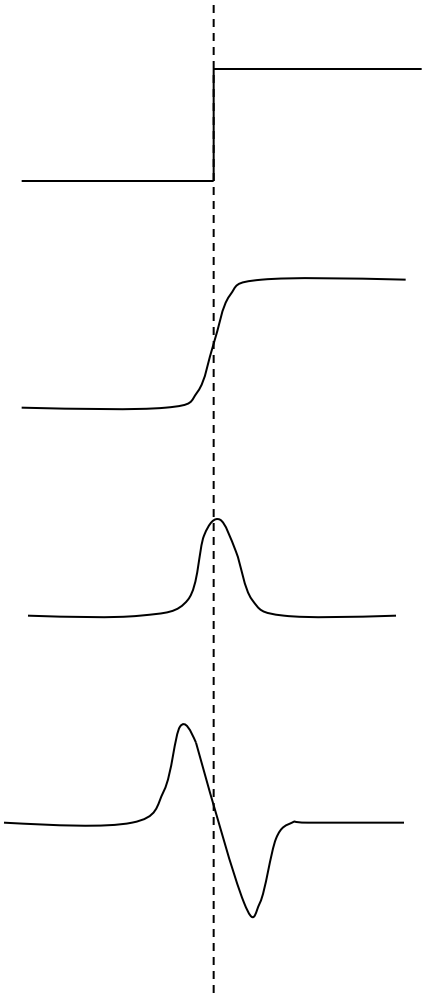
Characterization of Edges

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

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

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

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

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

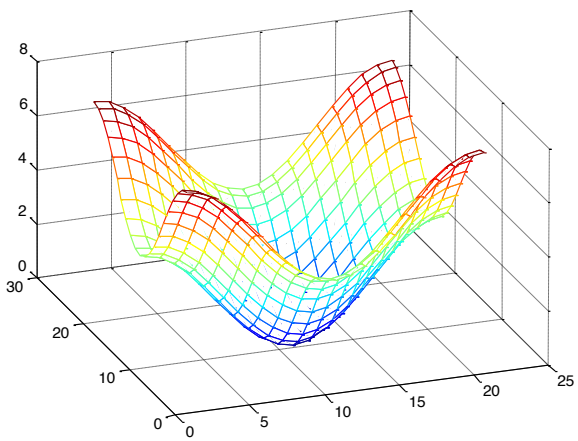
$$\nabla_f^2 = \left[f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)\right] - 4f(x, y)$$

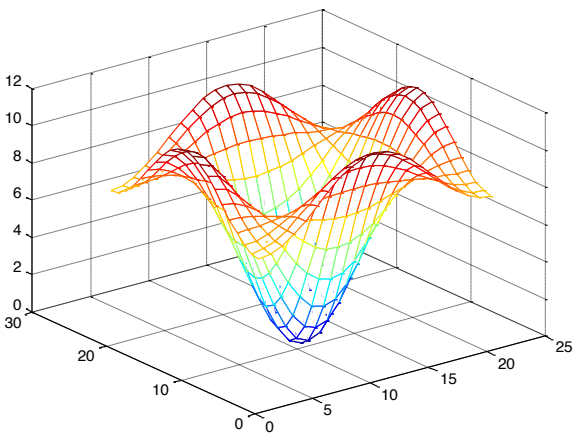
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix};$$

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

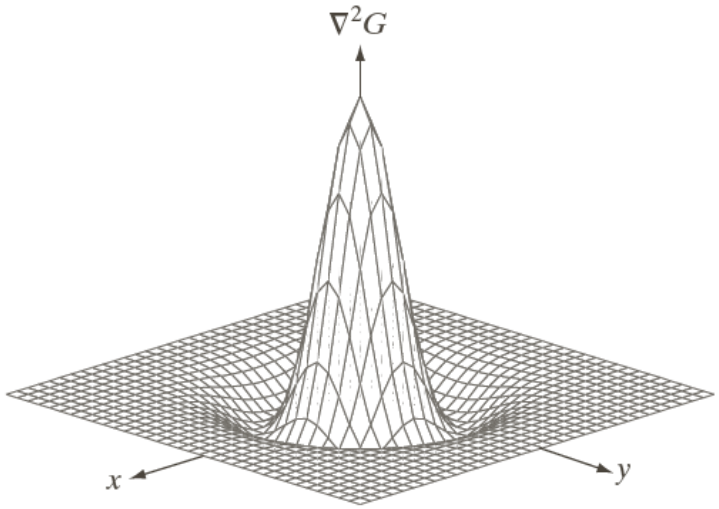
- Continuous form

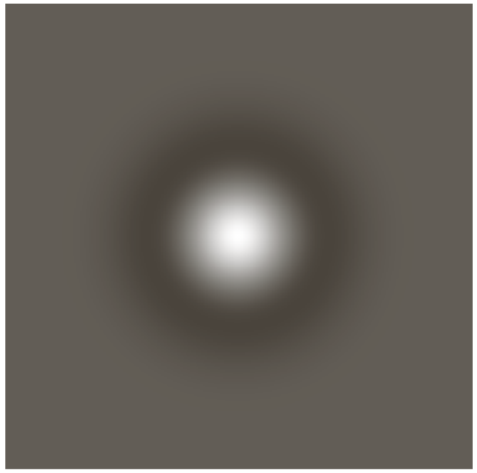
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

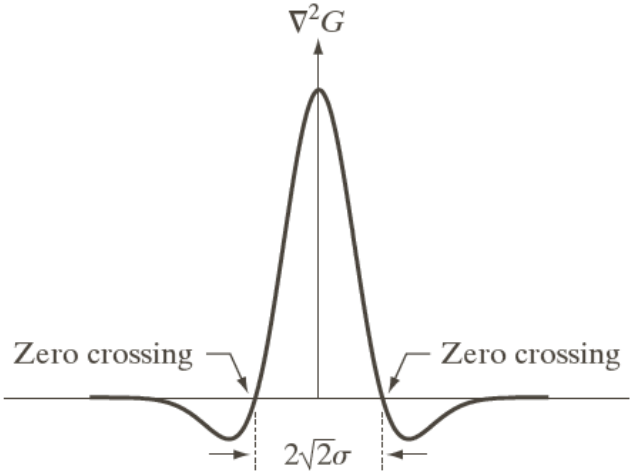
$$\nabla^2 G(x,y) = \frac{\partial^2 G}{\partial^2 x} + \frac{\partial^2 G}{\partial^2 y} = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

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

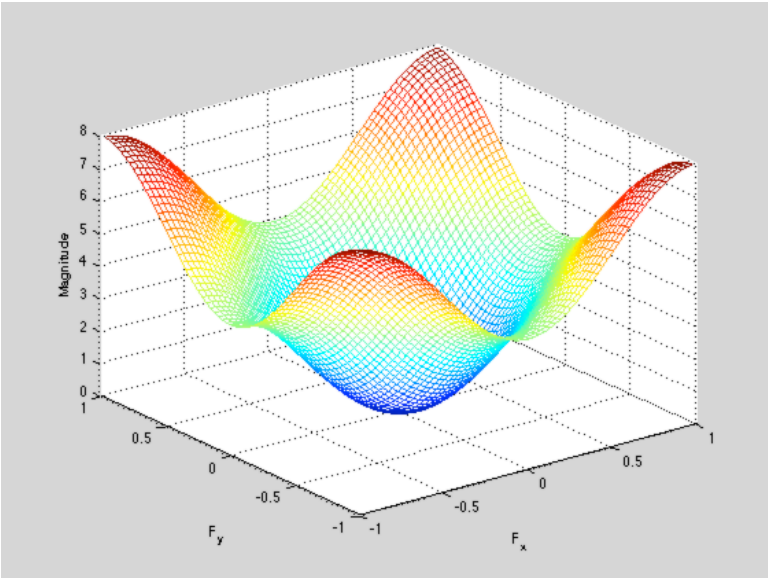
FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

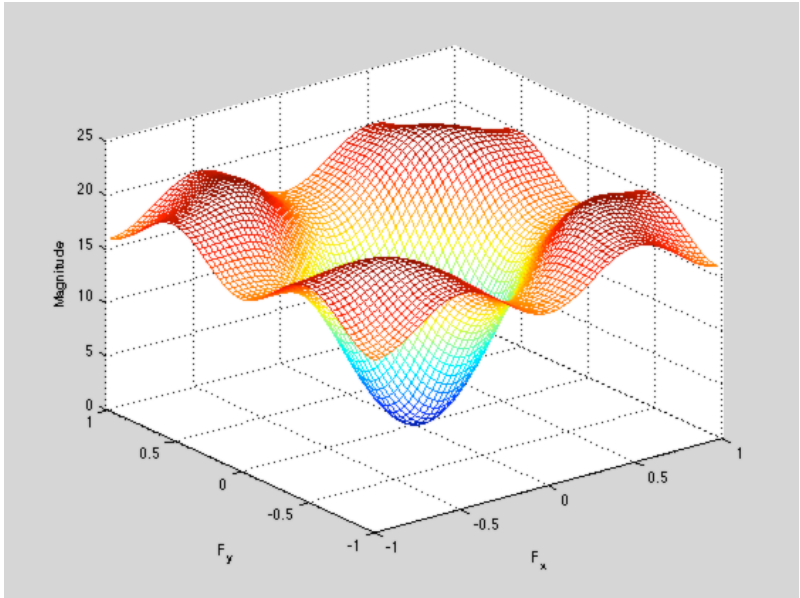
Laplacian vs. LoG in Freq. Domain

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

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

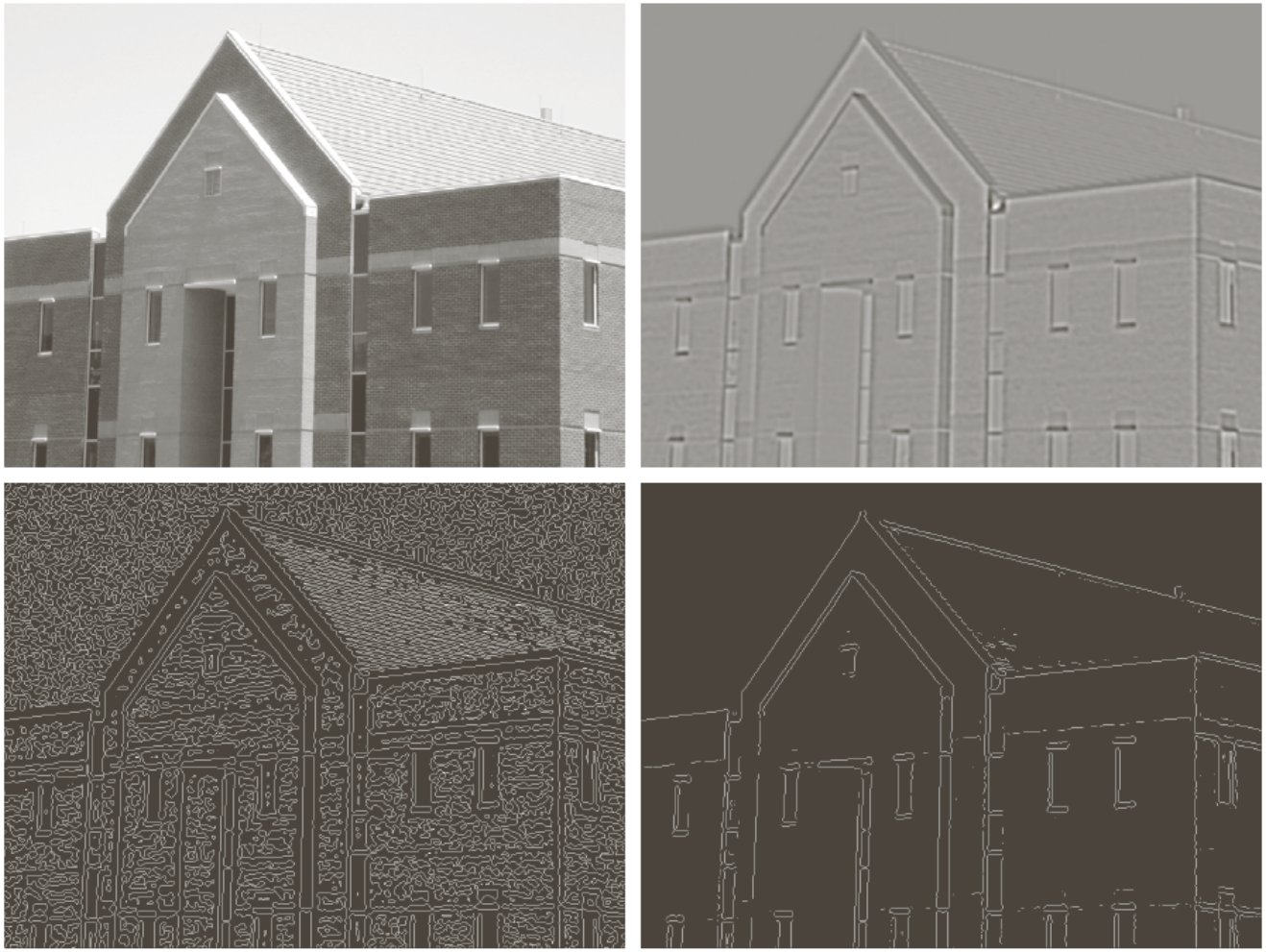
LOG filtered

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.

Pros and Cons

- Can locate edges more accurately
- Can detect edges in various direction
- But more prone to noise
- Remedy:
  - Smoothing before applying Laplacian

Summary of Edge Detection Method

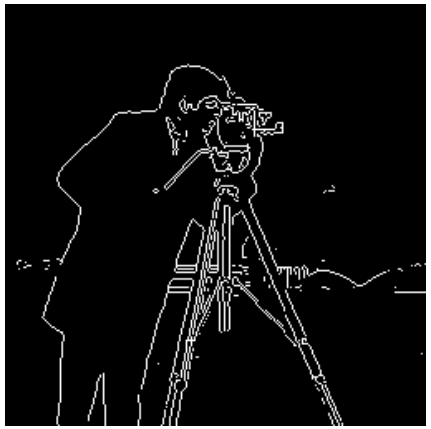
- First order gradient based:
  - Using edge detectors in two orthogonal directions
    - For each direction: low-pass along edge, band-pass across edge
  - Using edge detectors in multiple (>2) directions
  - Use threshold or detect local maximum across the edge direction
- Second order gradient based
  - 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

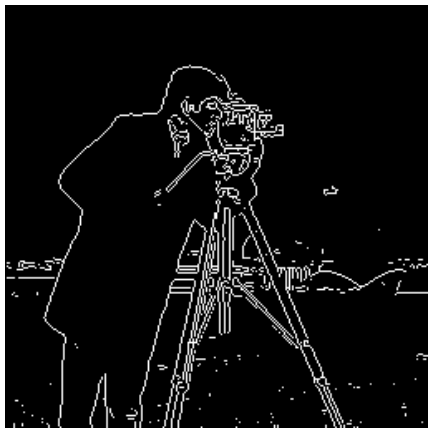
Sobel,T=0.14



LOG, T=0.0051

canny,T=[0.0313,0.0781]

Sobel,T=0.1



LOG, T=0.01

canny,T=[0.1,0.15]

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

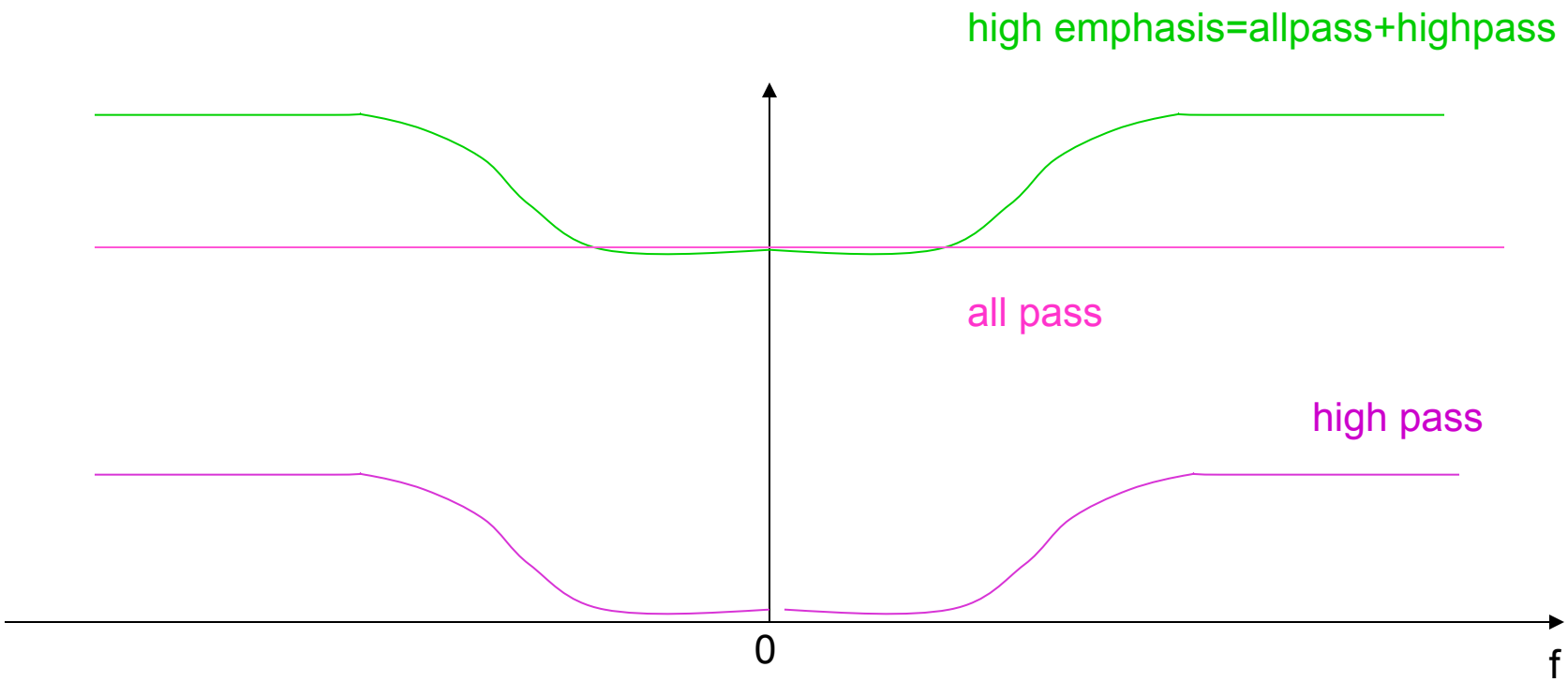
Designing Sharpening Filter Using High Pass Filters

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

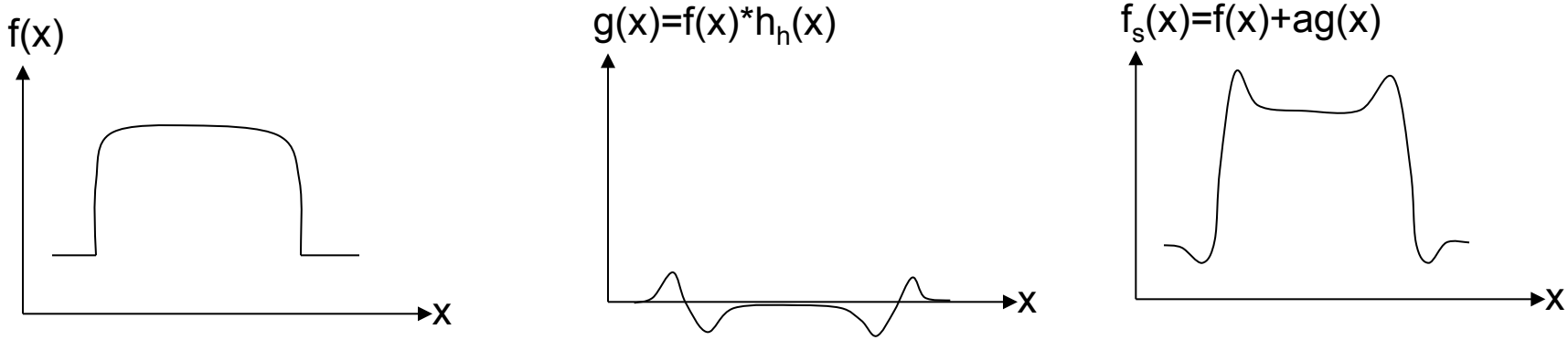
$$f_s = f + \lambda f_h$$

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

Interpretation in Freq Domain



High Emphasis Filter Using Laplacian Operator as Highpass Filter



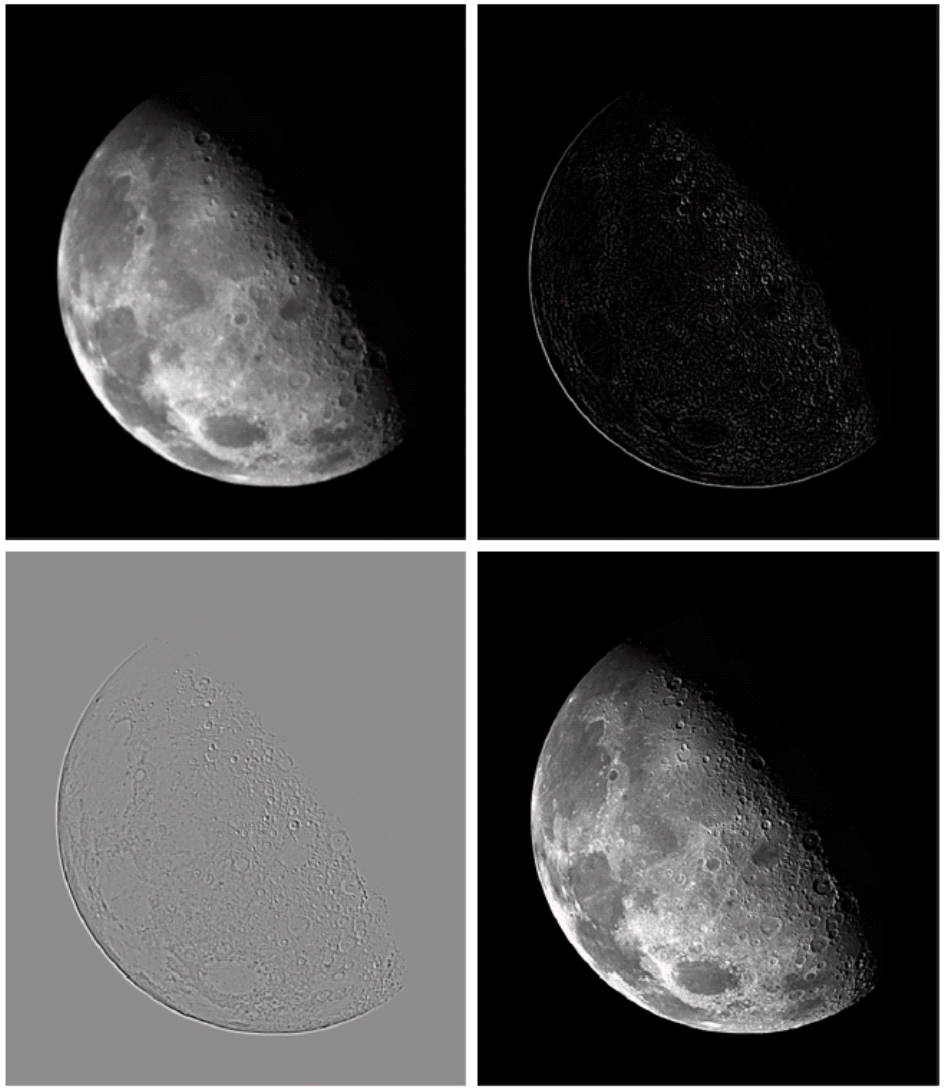
$$H_h = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \Rightarrow H_s = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad with \quad \lambda = 1.$$

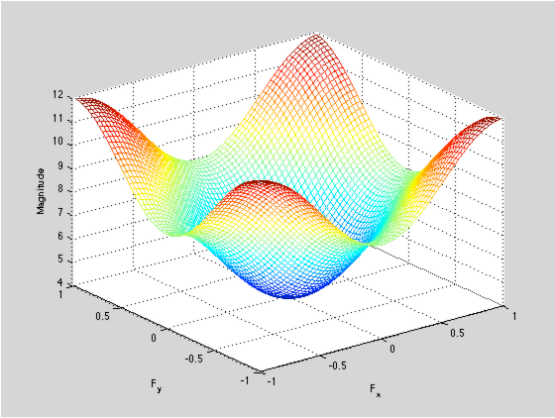
$$H_h = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \Rightarrow H_s = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 16 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad with \quad \lambda = 1.$$

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.)





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

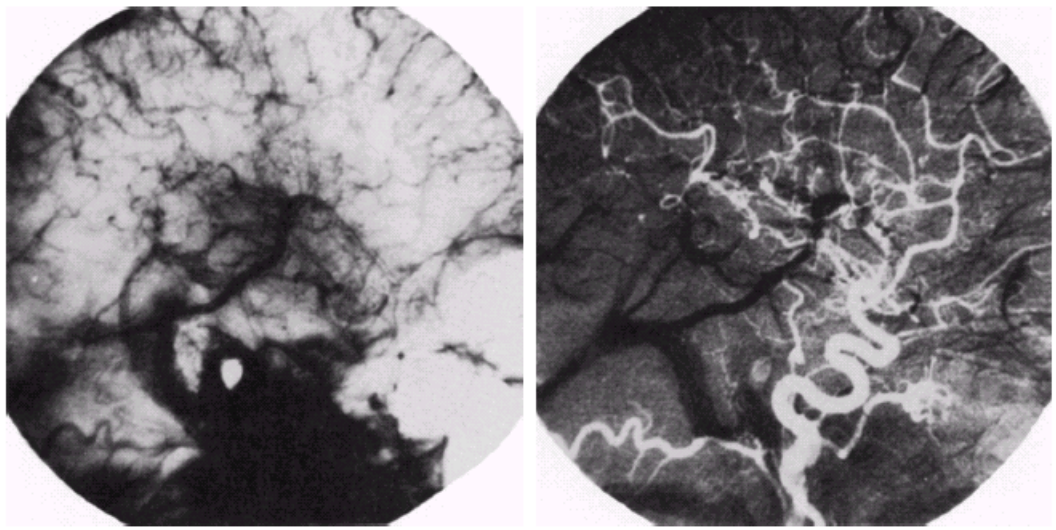
Challenges of Noise Removal and Image Sharpening

- How to smooth the noise without blurring the details too much?
- How to enhance edges without amplifying noise?
- Still a active research area
  - Wavelet domain processing

Wavelet-Domain Filtering

Courtesy of Ivan Selesnick

Feature Enhancement by Subtraction



a b

FIGURE 3.29 Enhancement by image subtraction. (a) Mask image. (b) An image (taken after injection of a contrast medium into the bloodstream) with mask subtracted out.

Taking an image without injecting a contrast agent first. Then take the image again after the organ is injected some special contrast agent (which go into the bloodstreams only). Then subtract the two images --- A popular technique in medical imaging

Non-Linear Filters

- Non-linear:
  - $T(f_1 + f_2) \neq T(f_1) + T(f_2)$
  - $T(af) \neq a\,T(f)$
- Median filter
- Rank-order filter (median is a special case)
- Morphological filter

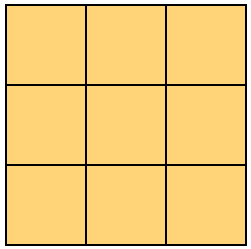
Median Filter

- Problem with Averaging Filter
  - Blur edges and details in an image
  - Not effective for impulse noise (Salt-and-pepper)
- Median filter:
  - Taking the median value instead of the average or weighted average of pixels in the window
    - Sort all the pixels in an increasing order, take the middle one
  - The window shape does not need to be a square
  - Special shapes can preserve line structures

Median Filter: 3x3 Square Window

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

Window shape



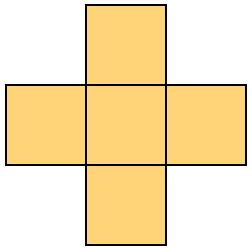
100	100	100	100	100
100	100	200	100	100
100	200	200	200	100
100	100	195	100	100
100	100	100	100	100

Matlab command: medfilt2(A,[3 3])

Median Filter: 3x3 Cross Window

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

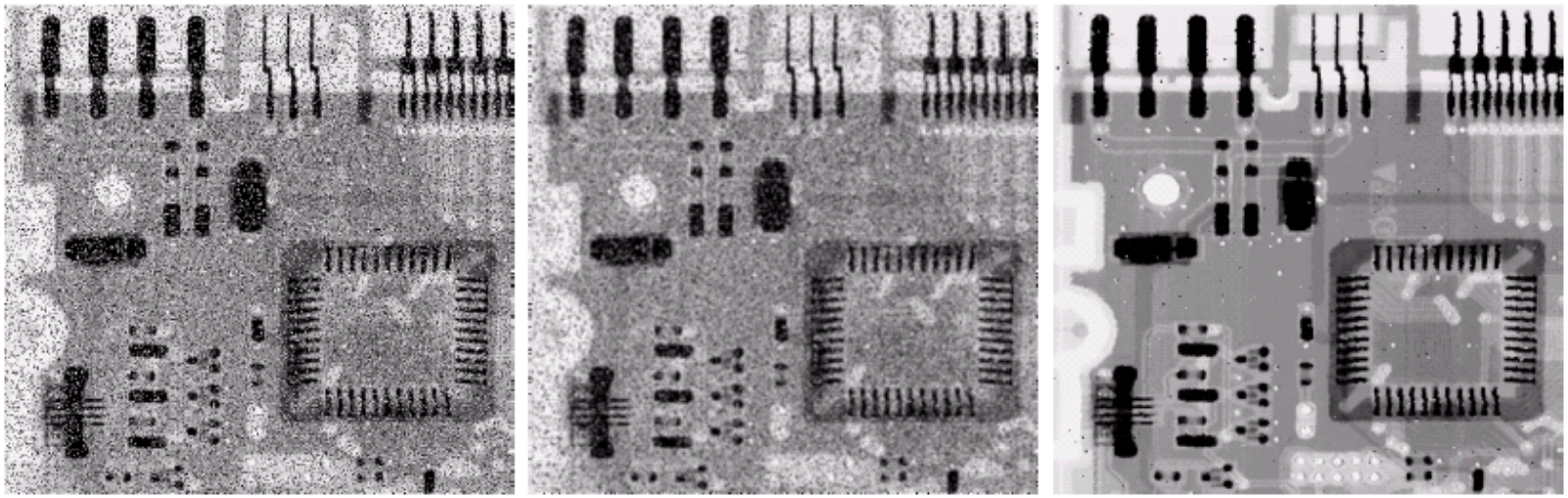
Window shape



100	100	100	100	100
100	195	200	200	100
100	200	200	200	100
100	195	200	195	100
100	100	100	100	100

Note that the edges of the center square are better reserved

Example



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Rank order filters

- Rank order filters
  - Instead of taking the mean, rank all pixel values in the window, take the n-th order value.
  - E.g. max or min or median

Morphological Processing

- Morphological operations are originally developed for bilevel images for shape and structural manipulations.
- Basic functions are *dilation* and *erosion*.
- Concatenation of dilation and erosion in different orders result in more high level operations, including *closing* and *opening*.
- Morphological operations can be used for smoothing or edge detection or extraction of other features.
- Belongs to the category of spatial domain filter.
- Gray-scale morphological filters are non-linear filters.
- Morphological filters are powerful tools and can accomplish various desired tasks: noise removal, edge detection, image smoothing, bi-level image shape smoothing.
- Self-study

Summary

- Noise removal using low-pass filters (averaging, Gaussian).
- Edge detection
  - Directional filters: Sobel, DoG (difference of Gaussian) + finding local maxima
  - Isotropic filters: 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 pass: coefficient sum=0
  - Sharpening: coefficient sum=1, has negative coefficients
- Median filter is particularly effective for removing impulse noises (salt-and-pepper)

Reading Assignments

- Image smoothing and sharpening
  - Gonzalez & Woods, "Digital Image Processing", Prentice Hall, 2008, Section 3.5, 3.6
  - Note: in Gonzalez & Woods, DSFT was not introduced, rather DFT
- Median filter and morphological filter
  - Gonzalez & Woods, "Digital Image Processing", Prentice Hall, 2008, Section 5.3. Chap. 9.

Written Assignment

1. For the three filters given below (assuming the origin is at the center): a) find their Fourier transforms (2D DTFT); 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 DTFT as a function of both u and v, using Matlab plotting function. c) Compare the functions of the three filters. In your calculation, you should make use of the separable property of the filter whenever appropriate. If necessary, split the filter into several additive terms such that each term can be calculated more efficiently.

$$H_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad H_2 = \frac{1}{24}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 12 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad H_3 = \frac{1}{24}\begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

2. Recall that the Laplacian of Gaussian filter is the Laplacian of a Gaussian function. Let the Gaussian function be given by $G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$. Show that the LOG filter can be written as

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

Write a Matlab program to generate the LoG filter by sampling the above analog form. Your program should allow you to specify the STD and filter size. Determine the filters for i) STD=1, filter size=7x7, and 2) STD=2, filter size=7x7. Plot their frequency responses (using freqz2( )). Compare the two filters in the spatial and frequency domain.

MATLAB Assignment (1)

1. Write a Matlab or C-program for implementing the following simple edge detection algorithm: For every pixel: i) find the horizontal and vertical gradients, g_x and g_y, using the Sobel operator; ii) calculate the magnitude of the gradient ; and iii) For a chosen threshold T, consider the pixel to be an edge pixel if $g_m \geq T$. Save the resulting edge map (Use a gray level of 255 to indicate an edge pixel, and a gray level of 0 for a non-edge pixel). Apply this program to your test image, and observe the resulting edge map with different T values, until you find a T that gives you a good result. Hand in your program and the edge maps generated by two different threshold values. Write down your observation. Hint: one automatic way to determine T is by sorting the pixels based on the gradient magnitudes, and choose T such that a certain percentage (say 5%) of pixels will be edge pixels. You can use either the matlab conv2( ) function or write your own code for the filtering part.

2. Write a program which can i) add salt-and-pepper noise to an image with a specified noise density, ii) perform median filtering with a specified window size. Consider only median filter with a square shape. Try two different noise density (0.05, 0.2) and for each density, comment on the effect of median filtering with different window sizes and experimentally determine the best window size. You can use imnoise( ) to generate noise. You should write your own function for median filtering. You can ignore the boundary problem by only performing the filtering for the pixels inside the valid region.

MATLAB Assignment (2)

3. In a previous assignment you have created a program for adding Gaussian noise and filtering using average filter. Apply the averaging filter and the median filter both to an image with Gaussian noise (with a chosen noise variance) and with salt-and-pepper noise (with a chosen noise density). Comment on the effectiveness of each filter on each type of noise.

4. (optional, no extra credits) Write a Matlab program to do edge detection using both the DoG and the LoG filter you designed. Your program should 1) Apply the two DoG filters and find the magnitude of the gradient at every pixel; 2) Apply the LoG filter; 3) for every pixel, check whether its gradient magnitude is greater than a threshold T. If yes, further check whether this pixel is a zero-crossing in the LOG filtered image. If yes, this pixel will be considered an edge pixel. Compare the resulting edge map with that obtained in 1).