Digital Image Processing
CS390S

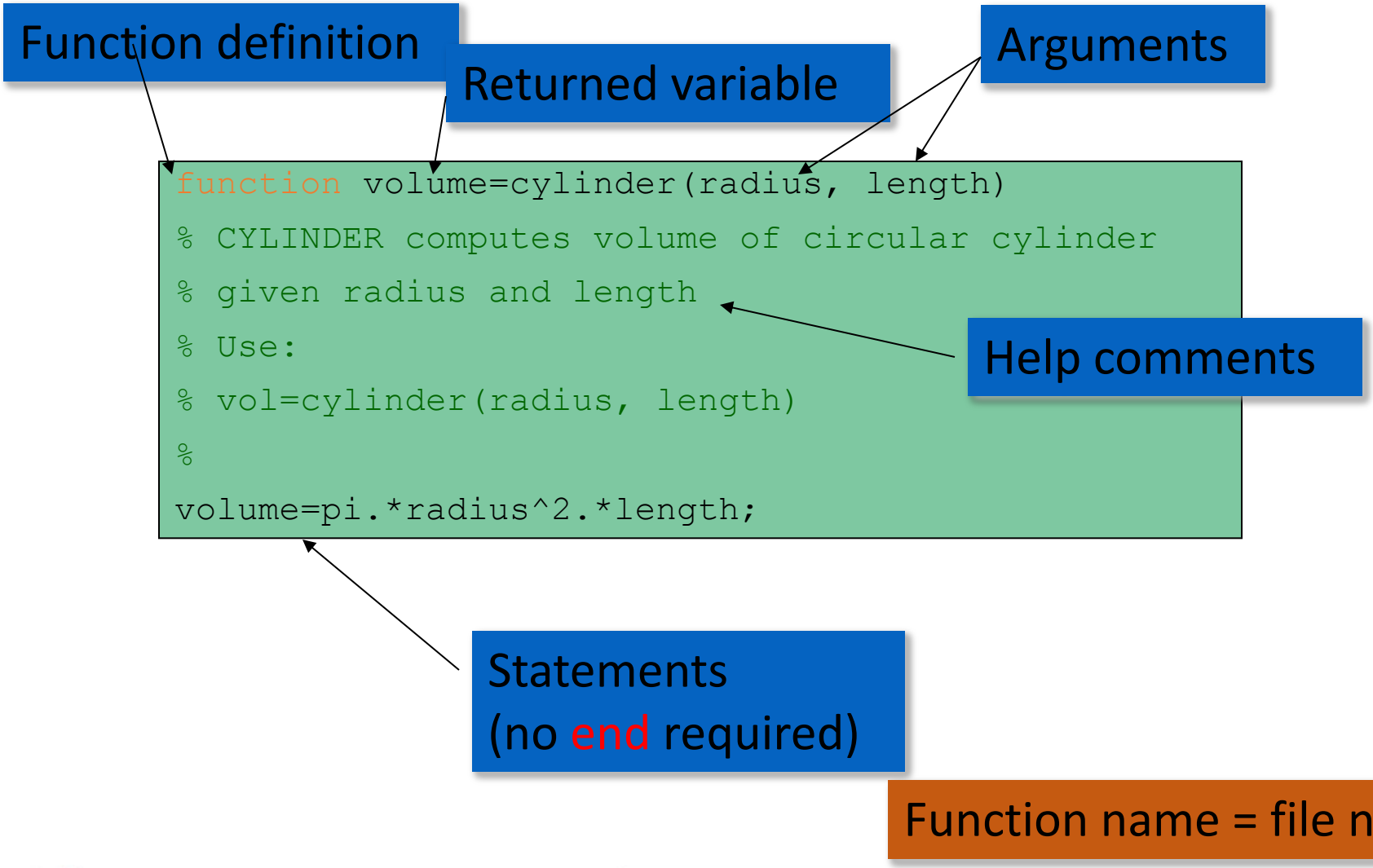
Feng Jiang
2018 Spring

METROPOLITAN STATE UNIVERSITY℠
OF DENVER

Day 6

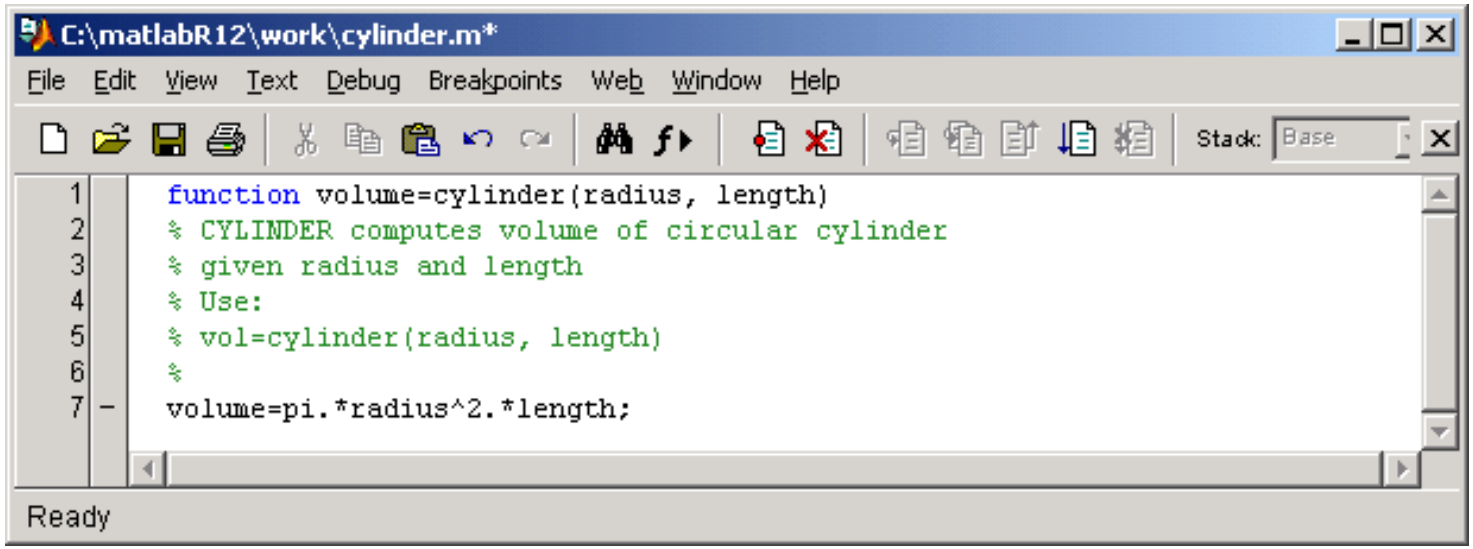
- Review derivative based filtering
- Other filters
- M function
- Edge detection
- Hough transform (straight line detection)
- Filtering in frequency domain

M-function


Function definition
Returned variable
Arguments
function volume=cylinder(radius, length)
% CYLINDER computes volume of circular cylinder
% given radius and length
% Use:
% vol=cylinder(radius, length)
%
volume=pi.*radius^2.*length;
Help comments
Statements
(no end required)
Function name = file name

Using M-File Editor to Create a Function

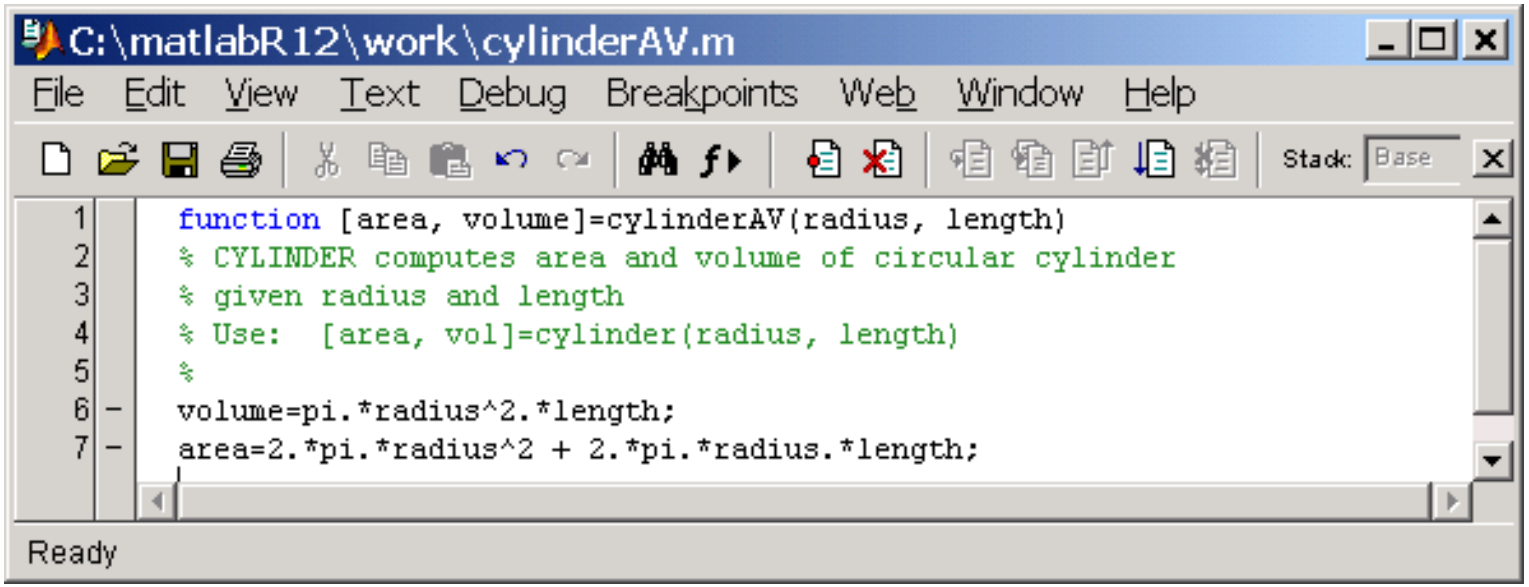
- From Command menu choose File/New/M-file
  - new edit window will appear with "Untitled"
  - enter code for your new function
  - when saving, make sure file name is same as function name (the file name is what Matlab uses to identify your m-functions)



On Your Own:
Figure out what all the options in the File, Edit, View & Text menus do…

M-function

- Return multiple variables



M-function

- Comments immediately following function statement will be listed by the Matlab **`Help`** function
  - Place comments that define what your function does
  - Include comments to illustrate how to use function
  - Include version, date and author

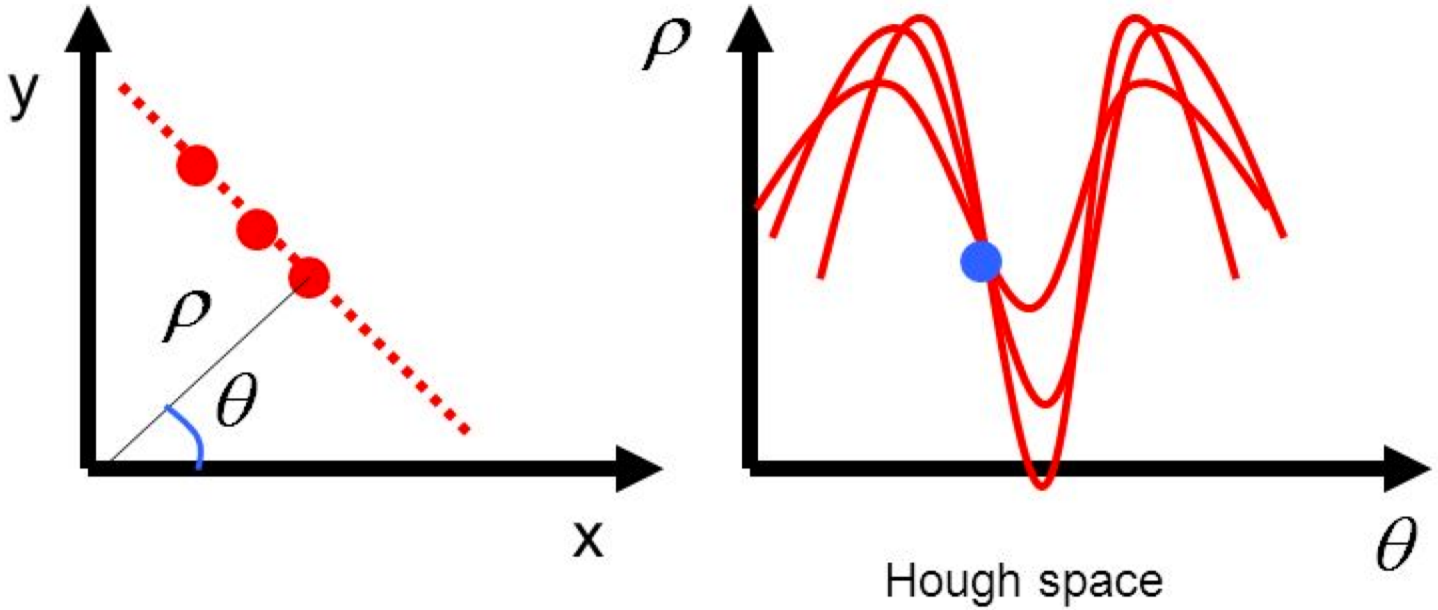
```
>> help cylinder


  CYLINDER computes volume of circular cylinder given radius and length
  Use:
  vol=cylinder(radius, length)


>>
```

Hough transform

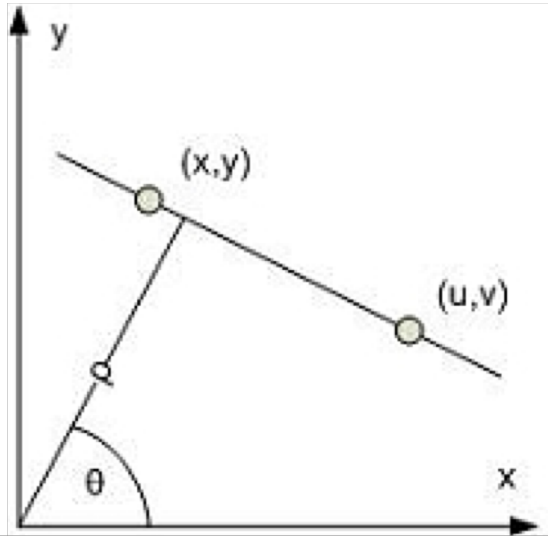
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

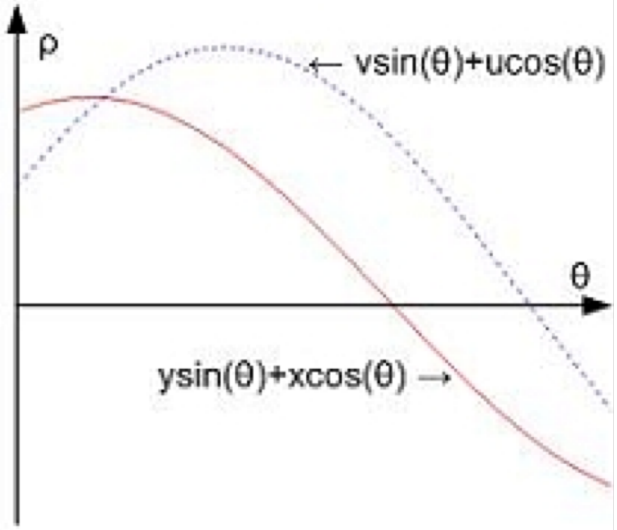
Issue : parameter space [m,b] is unbounded…

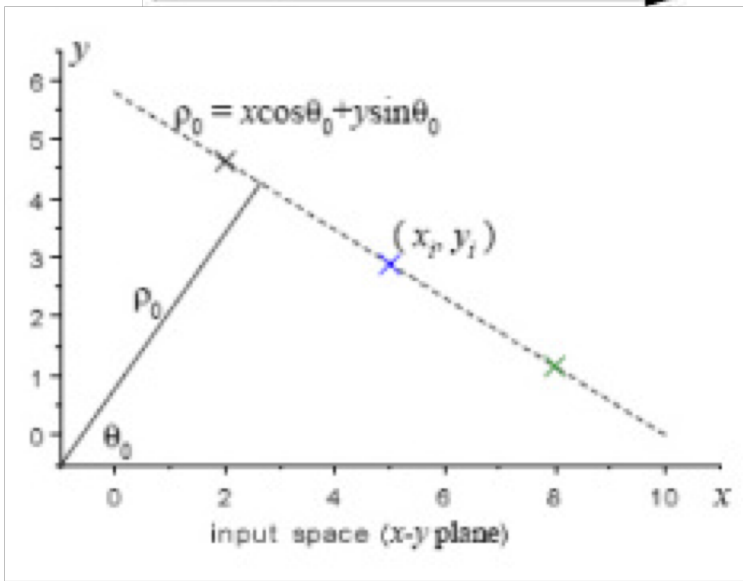
Use a polar representation for the parameter space

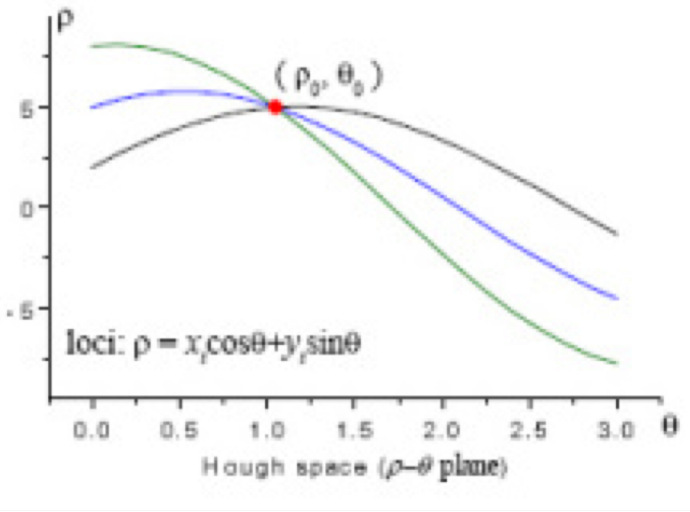


$$x\cos\theta + y\sin\theta = \rho$$

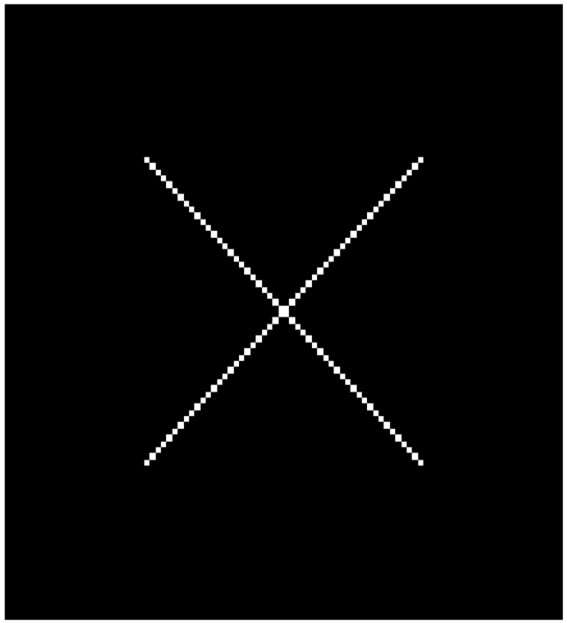


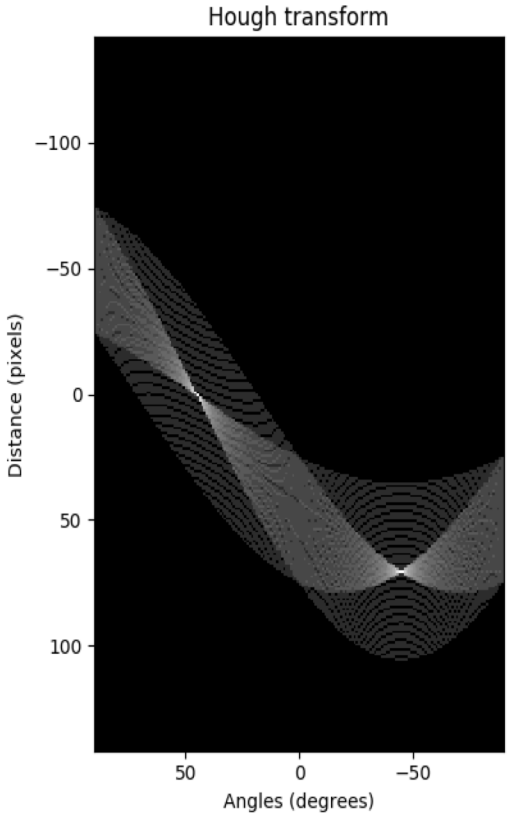


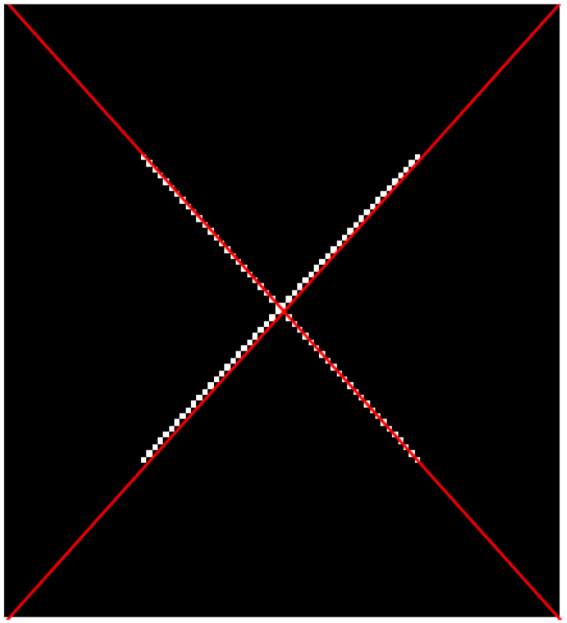


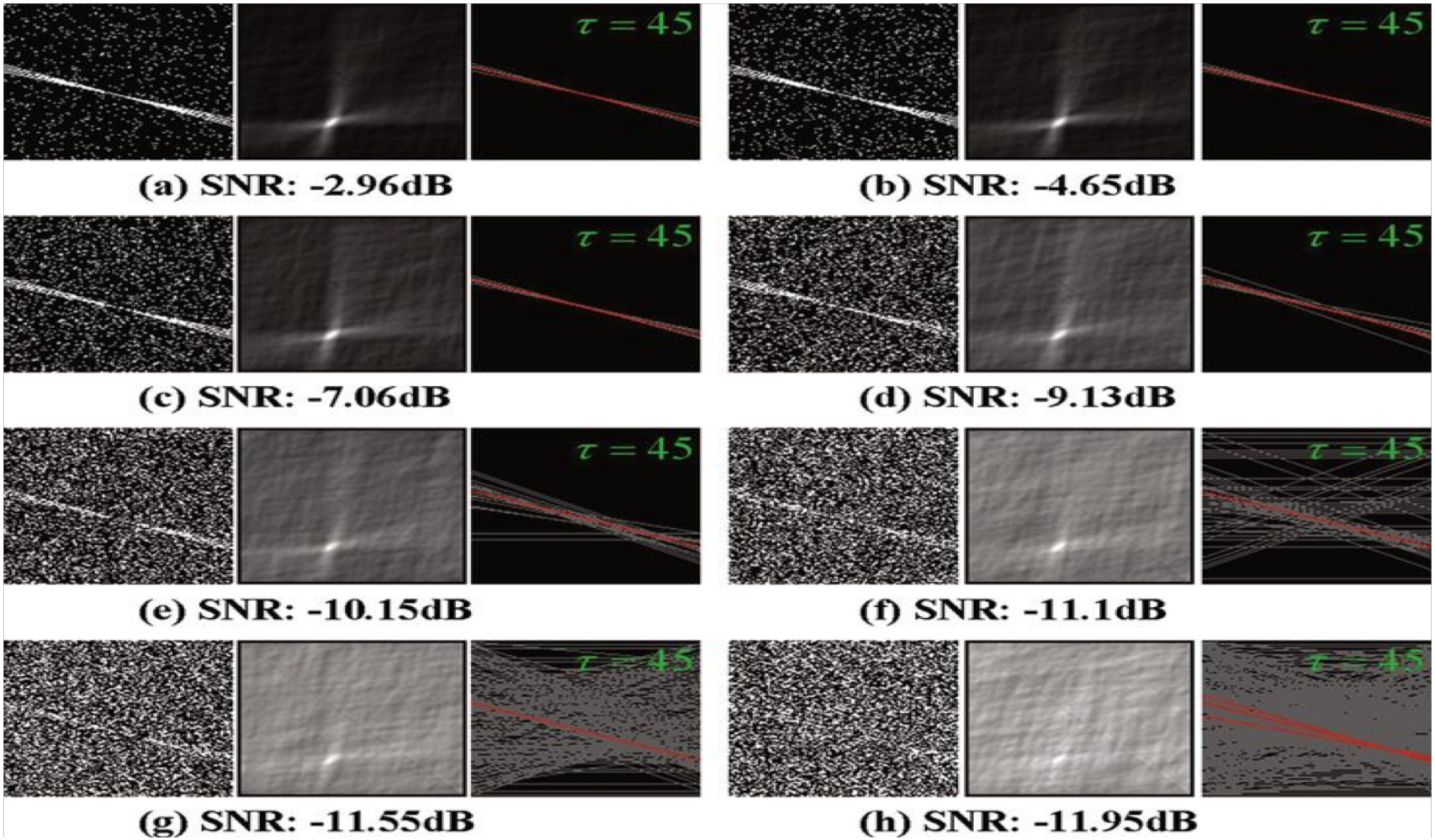


http://homepages.inf.ed.ac.uk/amos/hough.html


Input image


Hough transform
−100
−50
0
50
100
Distance (pixels)
50
0
−50
Angles (degrees)


Detected lines


τ = 45
(a) SNR: -2.96dB
τ = 45
(b) SNR: -4.65dB
τ = 45
(c) SNR: -7.06dB
τ = 45
(d) SNR: -9.13dB
τ = 45
(e) SNR: -10.15dB
τ = 45
(f) SNR: -11.1dB
τ = 45
(g) SNR: -11.55dB
τ = 45
(h) SNR: -11.95dB

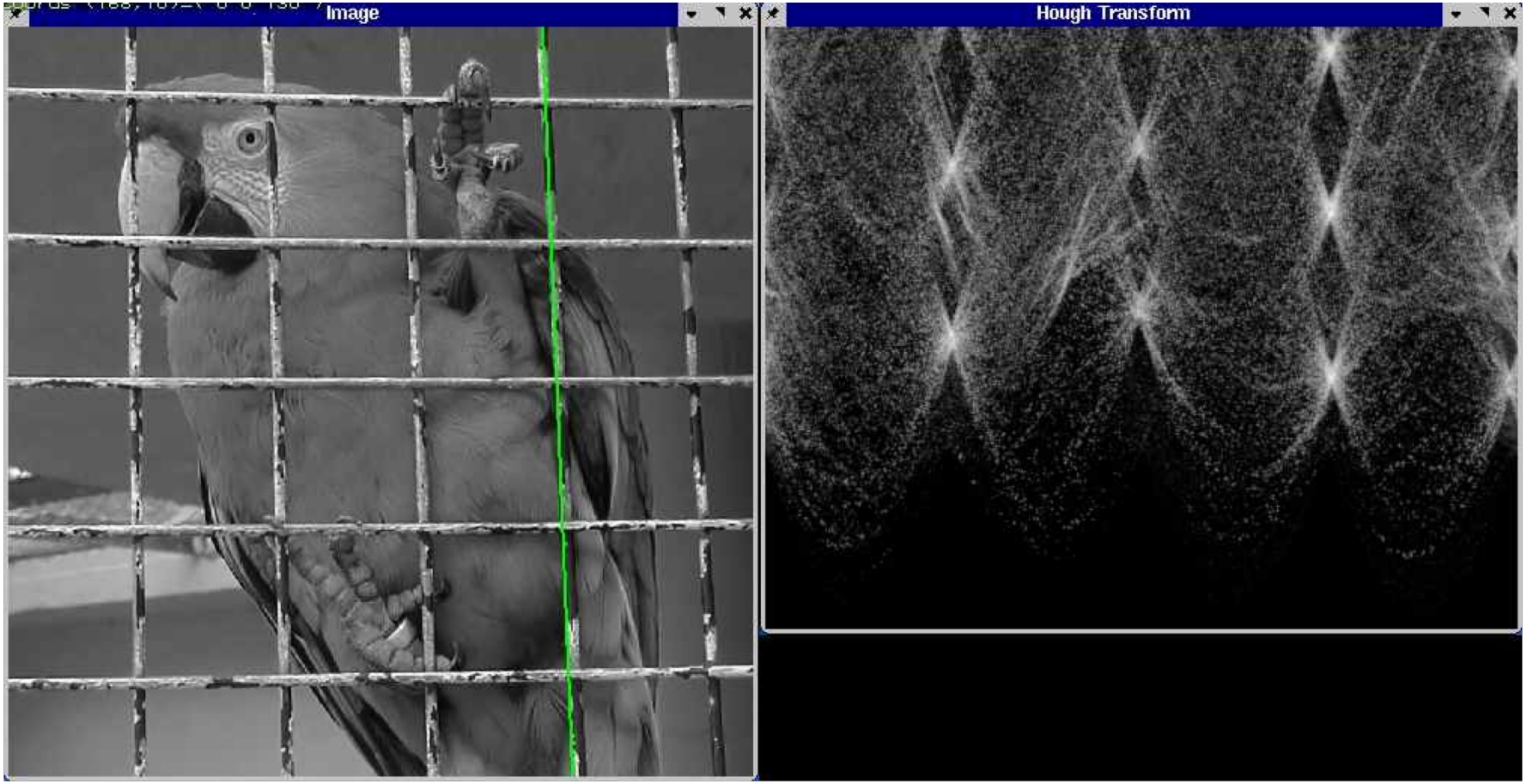
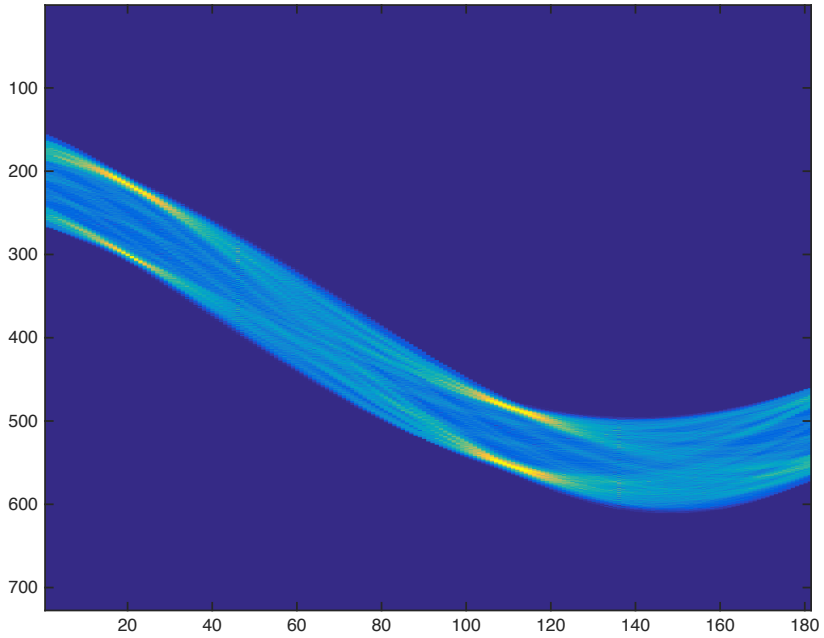
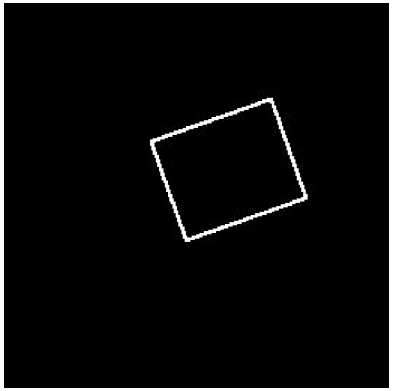


Image
Hough Transform

Exercise hough_show.m

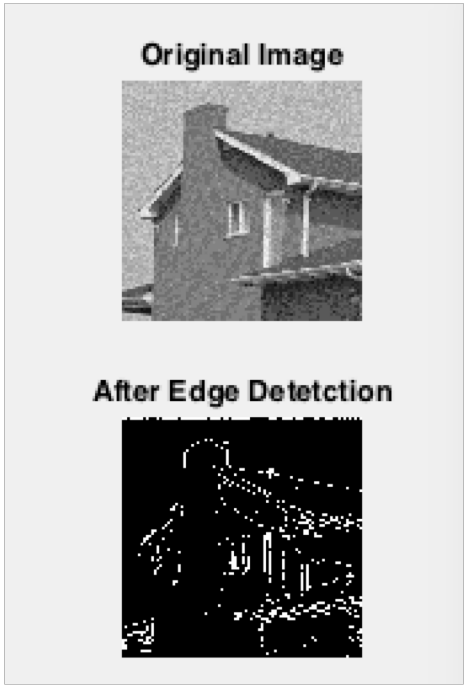


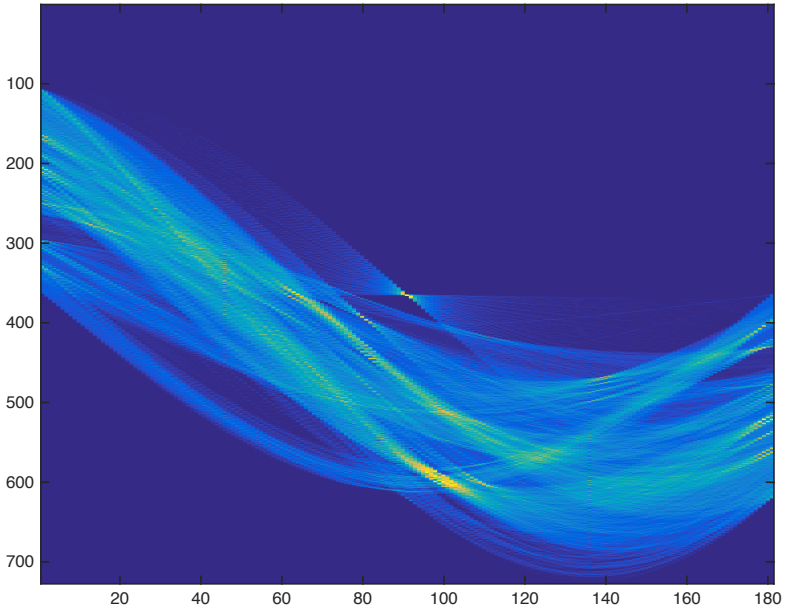
Original Image



After Hough Transform

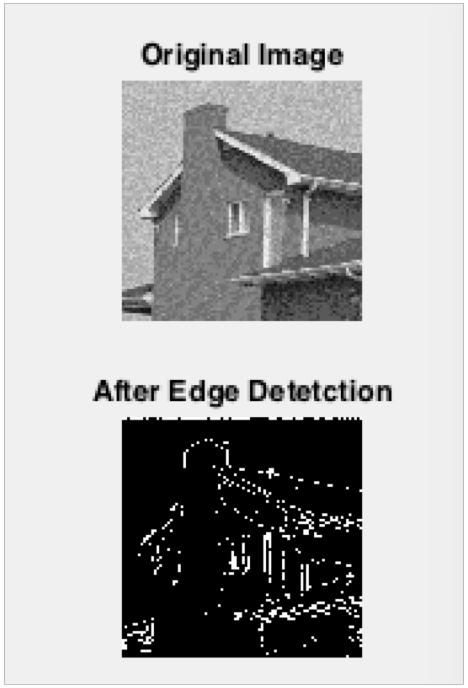
Exercise hough_show_edge.m

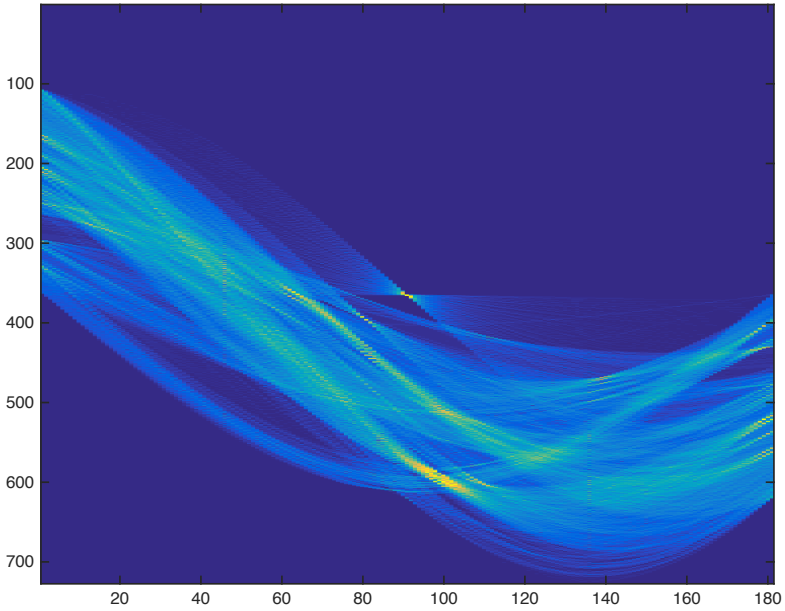




[Additional reading: http://laid.delanover.com/hough-transform-circle-detection-and-space-reduction/](http://laid.delanover.com/hough-transform-circle-detection-and-space-reduction/)

Exercise hough_show_edge.m





[Additional reading: http://laid.delanover.com/hough-transform-circle-detection-and-space-reduction/](http://laid.delanover.com/hough-transform-circle-detection-and-space-reduction/)

Edge detection

- Stanford slide

Edge detection

- Gradient-based edge operators
  - Prewitt
  - Sobel
  - Roberts
- Laplacian zero-crossings
- Canny edge detector

Edge detection

- Stanford slide
- Matlab funtions help: edge()

```
I = imread('circuit.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'canny');
imshow(BW1);
```

- BW = edge(I) takes an intensity or a binary image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

Exercise

- Exercise ***edge_show.m***

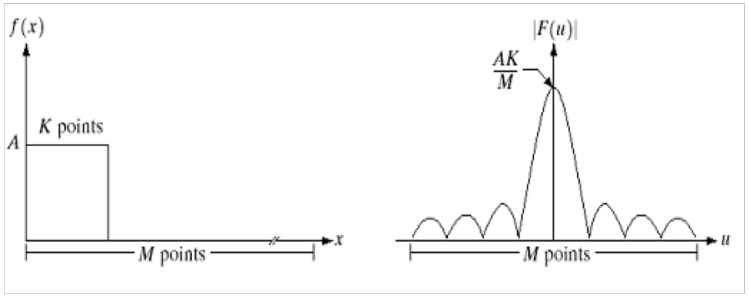
- Applying edge() to house.bmp
- Run sobel and canny filters
- Change different threshold values

Sobel

Canny

Filtering in frequency domain

- Fourier transform



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

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

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

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

Filtering in frequency domain

- discrete Fourier transform

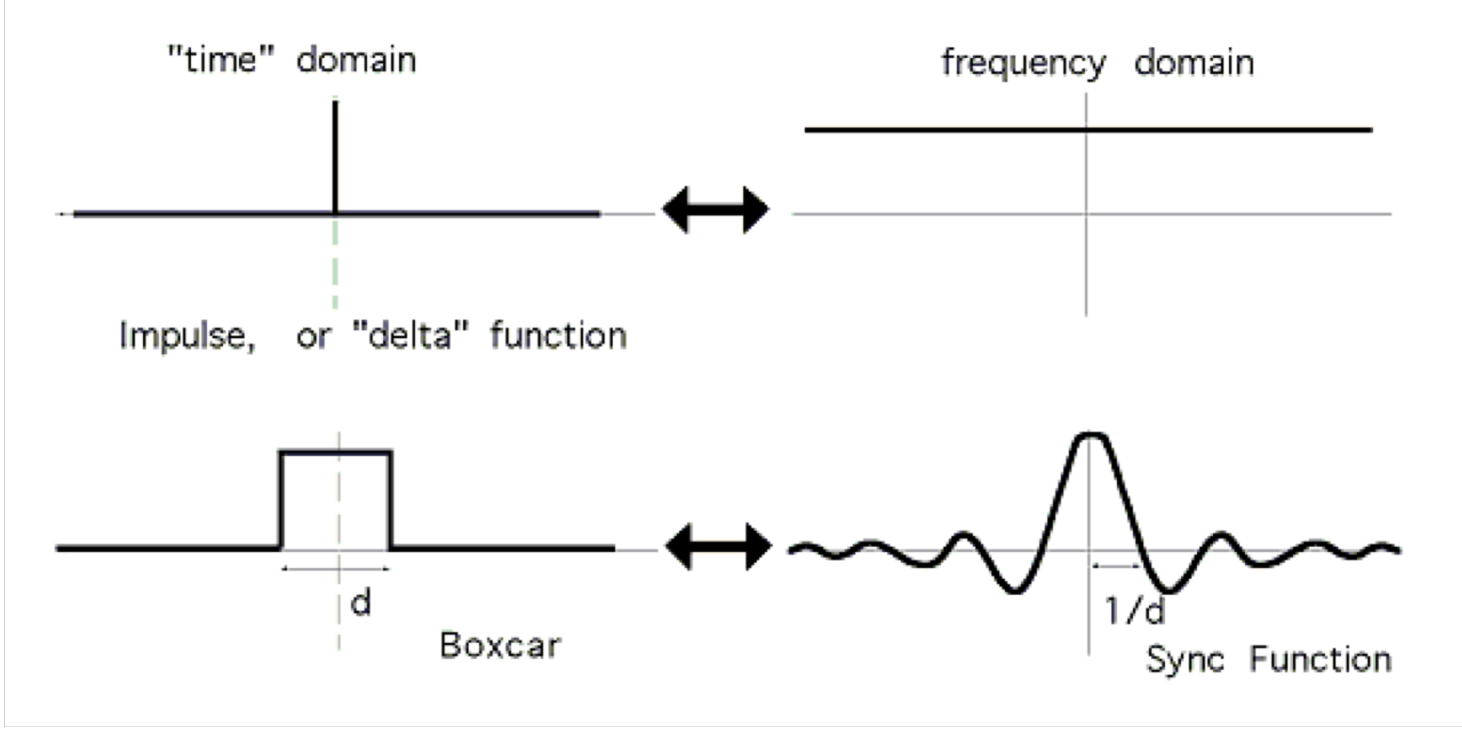
$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

- inverse of the above discrete Fourier transform

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Filtering in frequency domain

- Fourier transform



Filtering in frequency domain

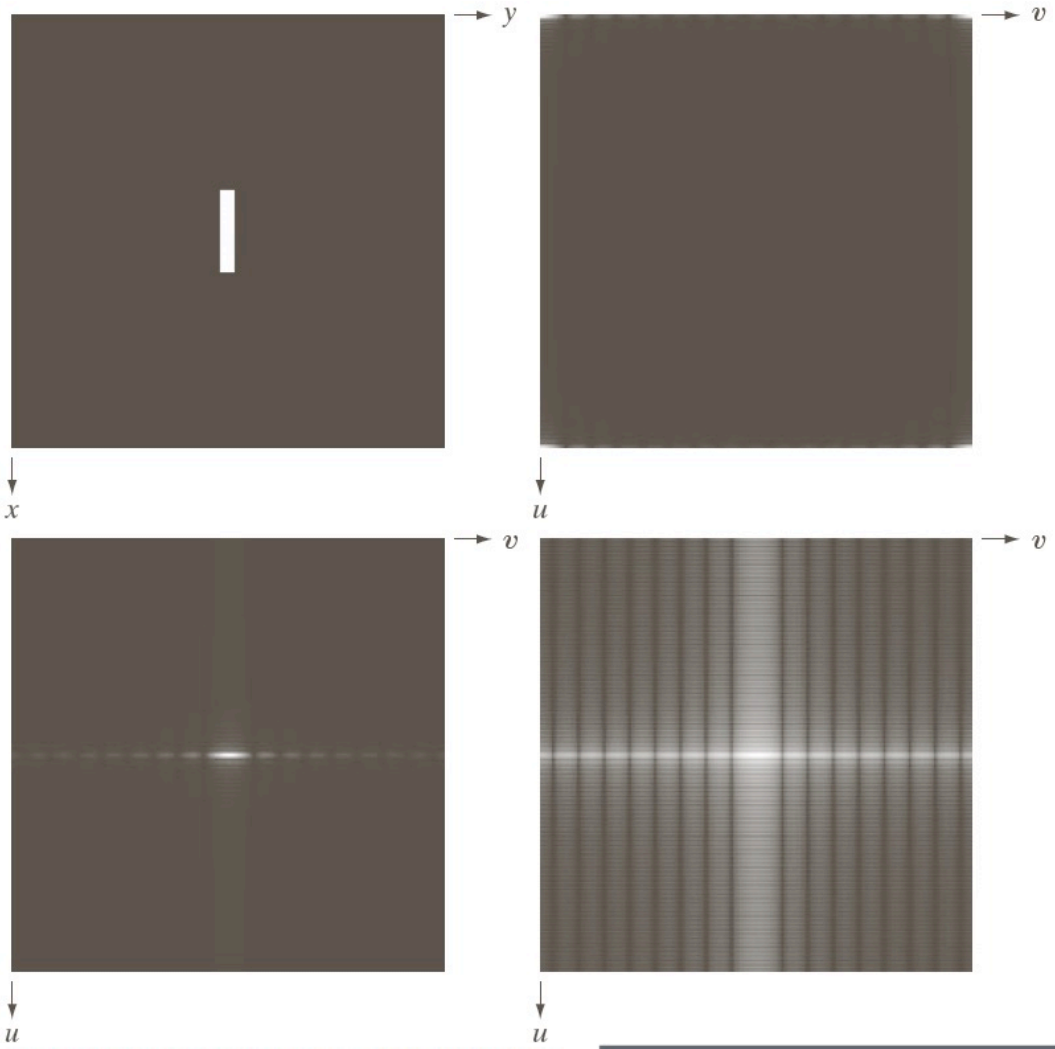
- Basic properties
  - u=v=0 → average gray level of an image
  - Low frequency → slowly varing components of an image
  - Higher frequency → faster gray level changes in the image

Filtering in frequency domain

- ***F(0,0)*** is equal to ***MN*** times the average value of ***f(x,y)***
- in MATLAB, ***F(0,0)*** is actually ***F(1,1)*** because array indices in MATLAB start at 1 rather than 0
- MATLAB has three functions to compute the DFT:
  - fft -for one dimension (useful for audio)
  - fft2 -for two dimensions (useful for images)
  - fftn -for n dimensions

- MATLAB has three related functions that compute the inverse DFT:
  - ifft
  - ifft2
  - ifftn

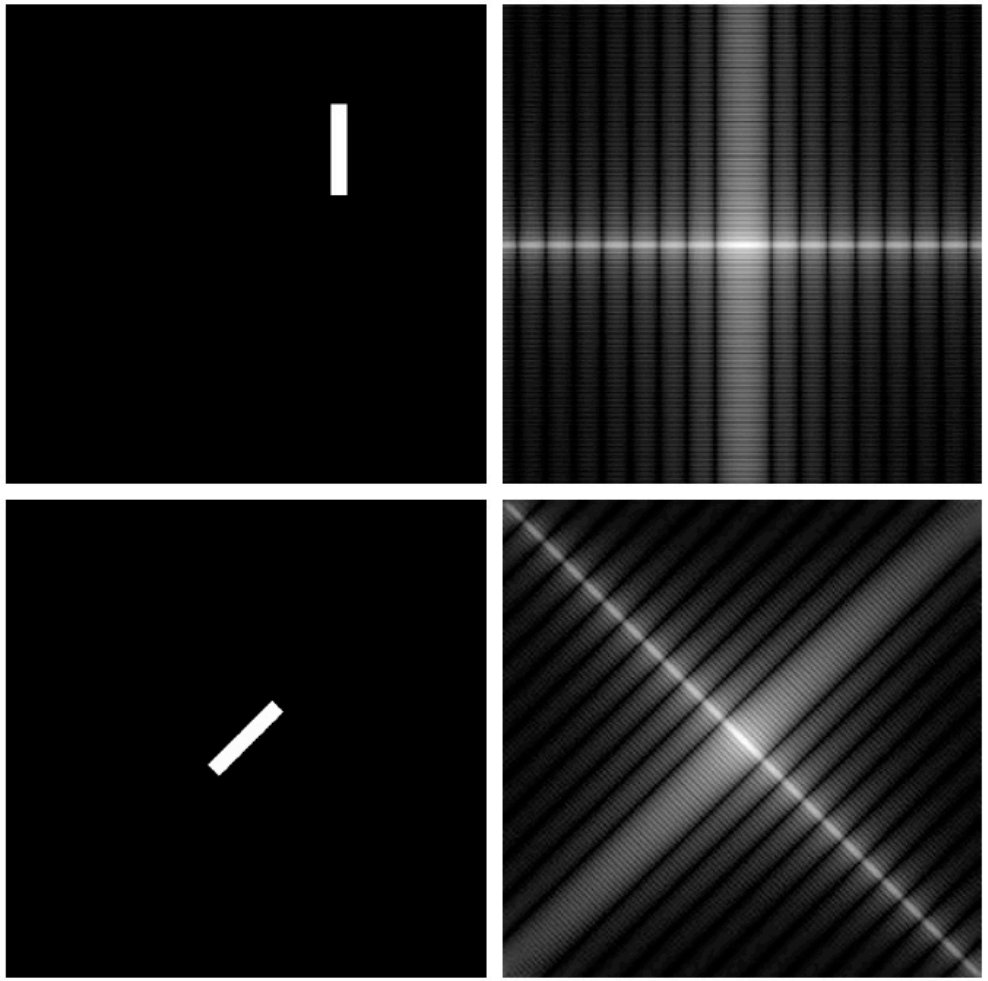
Filtering in frequency domain



a b
c d

FIGURE 4.24
(a) Image.
(b) Spectrum showing bright spots in the four corners.
(c) Centered spectrum. (d) Result showing increased detail after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The coordinate convention used throughout the book places the origin of the spatial and frequency domains at the top left.

Filtering in frequency domain



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).

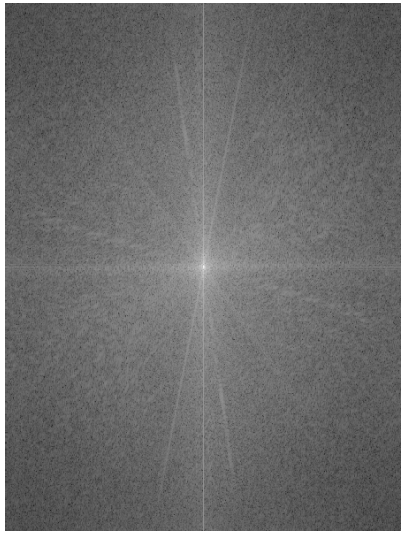
Filtering in frequency domain

- fft -for one dimension (useful for audio)
- fft2 -for two dimensions (useful for images)
- fftn -for n dimensions

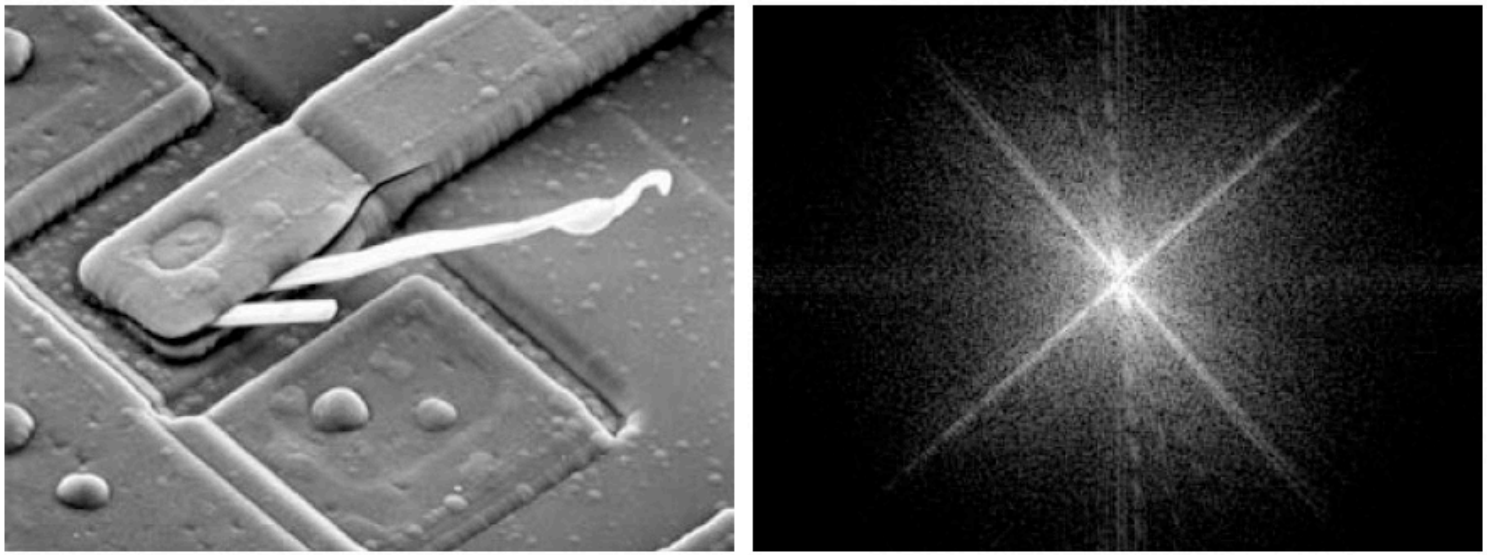
- ifft
- ifft2
- ifftn

original image

In the frequency domain



Filtering in frequency domain



a b

FIGURE 4.29 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

Filtering in frequency domain

- **convolution theorem** shows an interesting relationship between the spatial domain and frequency domain:

$$f(x,y)*h(x,y) \Leftrightarrow H(u,v)F(u,v)$$

Low pass and High pass filter

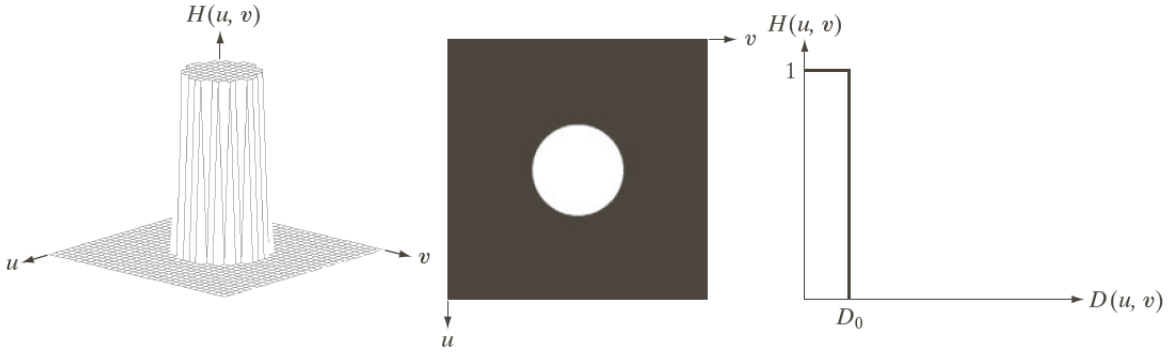
- Digital image processing
- Chapter 4.8.1 4.8.2 4.9.1 4.9.2

http://www.cs.uregina.ca/Links/class-info/425/Lab5/index.html

Filtering in frequency domain

- Ideal low pass filter

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



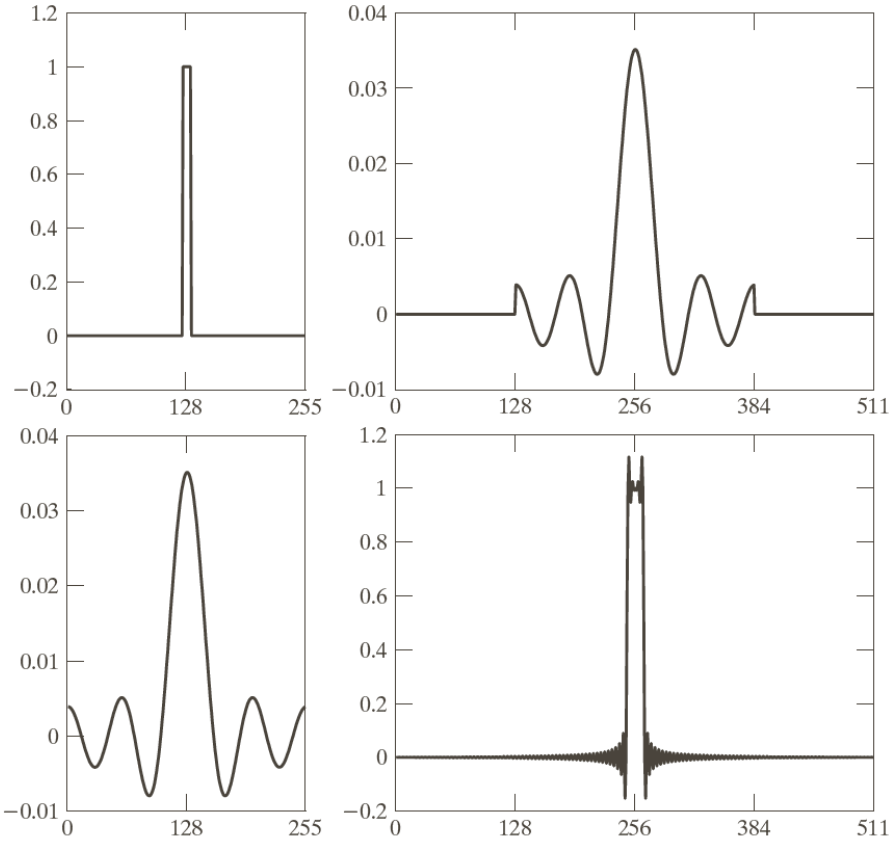
a b c

FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

$$D(u,v) = \left[(u - M/2)^2 + (v - N/2)^2\right]^{1/2}.$$

Filtering in frequency domain

- Ideal low pass filter

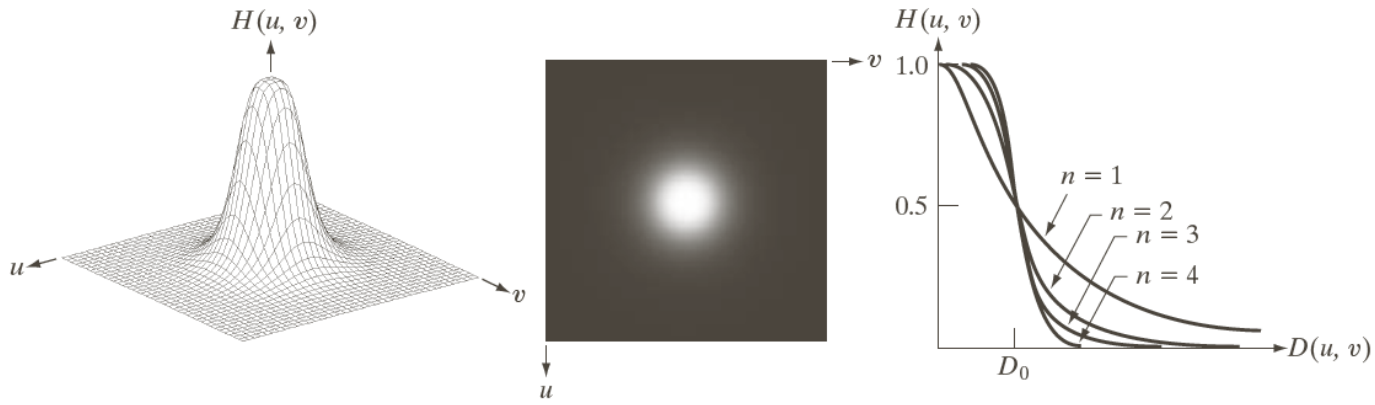


a c
b d

FIGURE 4.34
(a) Original filter specified in the (centered) frequency domain. (b) Spatial representation obtained by computing the IDFT of (a). (c) Result of padding (b) to twice its length (note the discontinuities). (d) Corresponding filter in the frequency domain obtained by computing the DFT of (c). Note the ringing caused by the discontinuities in (c). (The curves appear continuous because the points were joined to simplify visual analysis.)

Filtering in frequency domain

- Butterworth low pass filter



a b c

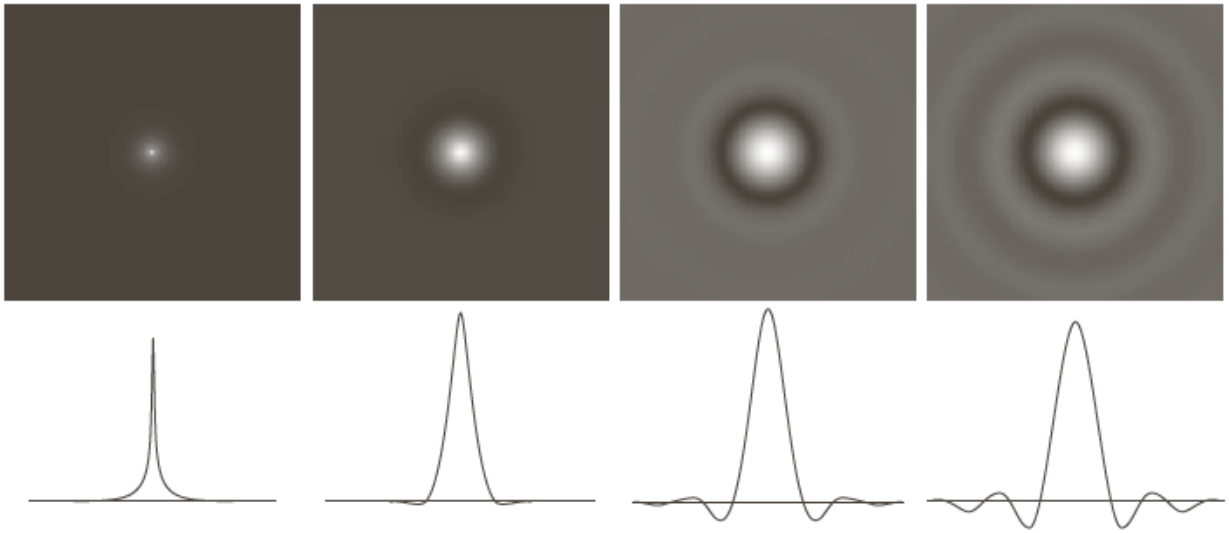
FIGURE 4.44 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

$$D(u, v) = \left[(u - M/2)^2 + (v - N/2)^2\right]^{1/2}.$$

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

Filtering in frequency domain

- Butterworth low pass filter

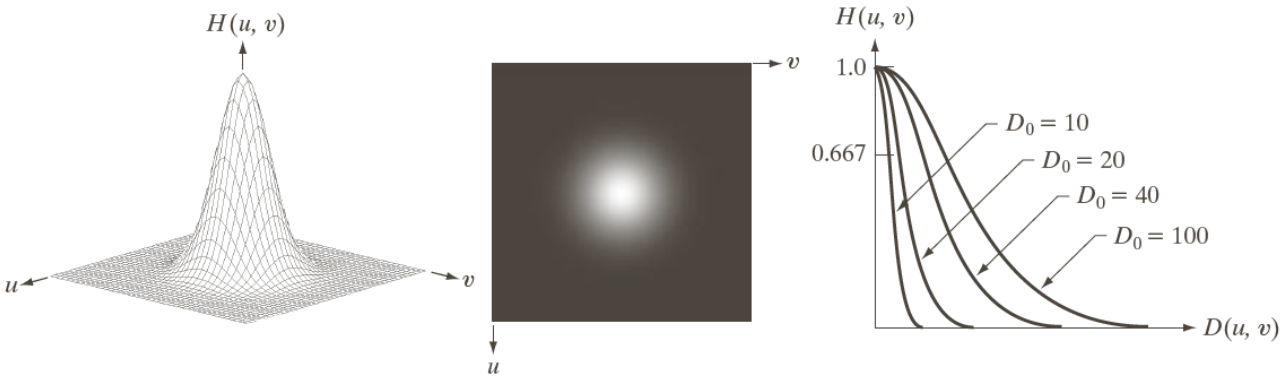


a b c d

FIGURE 4.46 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding intensity profiles through the center of the filters (the size in all cases is 1000 × 1000 and the cutoff frequency is 5). Observe how ringing increases as a function of filter order.

Filtering in frequency domain

- Gaussian low pass filter



a b c

FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

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

Filtering in frequency domain

- Gaussian low pass filter

a b c

FIGURE 4.50 (a) Original image (784 × 732 pixels). (b) Result of filtering using a GLPF with $D_0 = 100$. (c) Result of filtering using a GLPF with $D_0 = 80$. Note the reduction in fine skin lines in the magnified sections in (b) and (c).

Filtering in frequency domain

- Ideal high pass filter

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

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

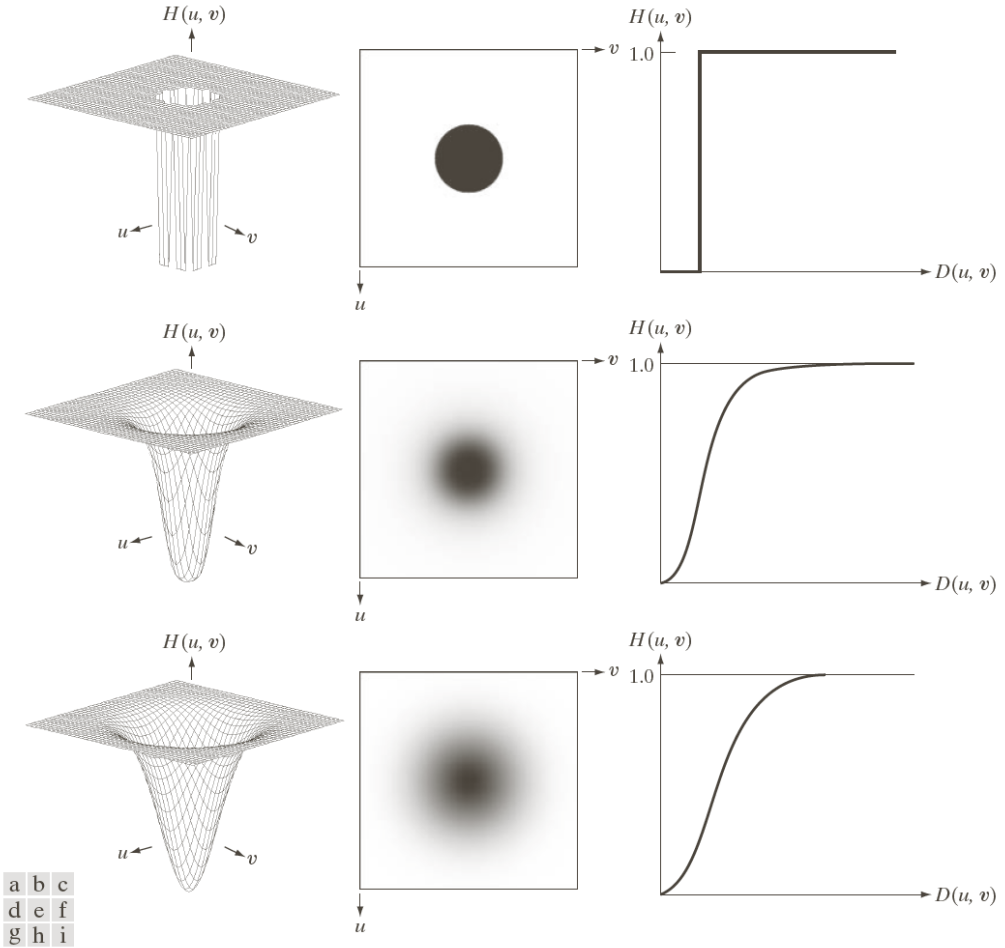
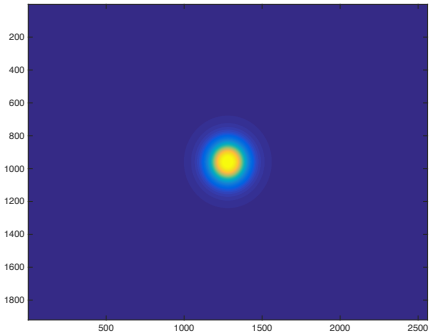
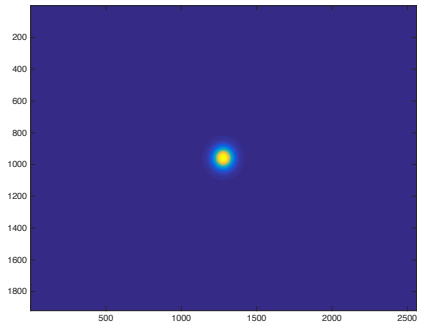


FIGURE 4.52 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Exercise: Low Pass High Pass

- LowPass.m
- HighPass.m
- LPass_HPass_show.m



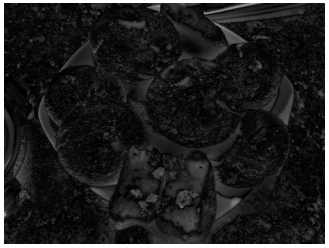


original image

original image

after lowpass d0 = 100

after highpass d0 = 5



after lowpass d0 = 50

after highpass d0 = 15