TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE
Faculty of Information Technology

Chapter 2.
Image Enhancement in Spatial Domain
(IMAGE PROCESSING)

PGS.TS. Hoàng Văn Dũng
Email: dunghv@hcmute.edu.vn

Outline

1. Background
2. Basic intensity transformation functions
3. Histogram processing
4. Fundamentals of spatial filtering
5. Smoothing (lowpass) spatial filters
6. Sharpening (highpass) spatial filters

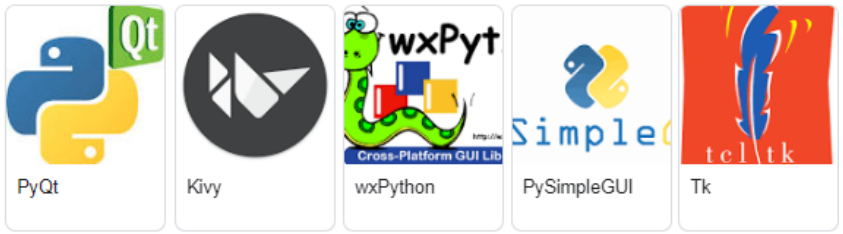
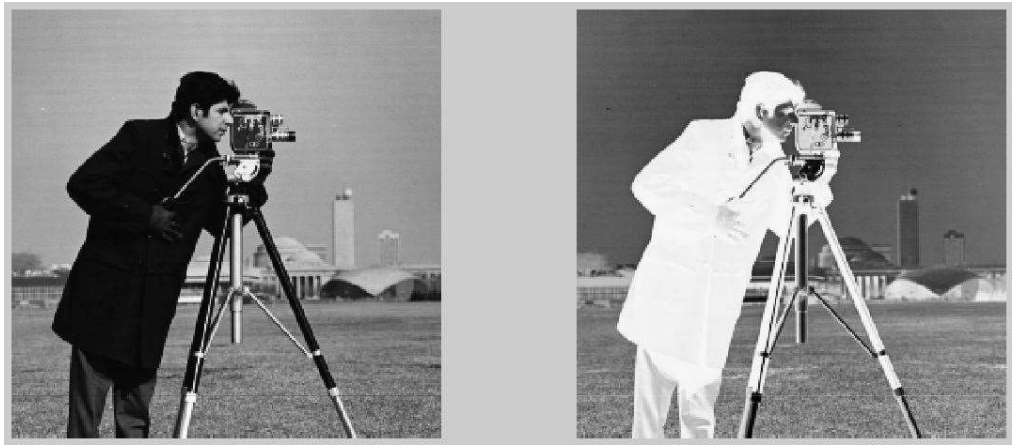


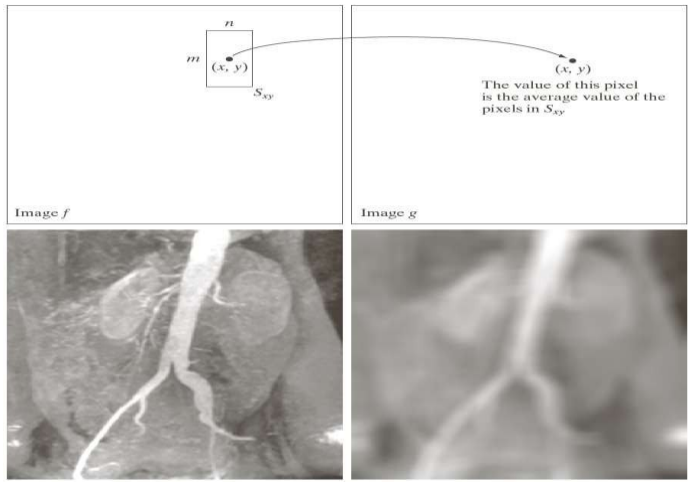
Image Enhancement

- The objective of image enhancement is to process an image so that the result is more *suitable* than the original image for a specific application.
- There are two main approaches:
  - Image enhancement in spatial domain: Direct manipulation of pixels in an image
    - Point processing: Change pixel intensities
    - Spatial filtering
  - Image enhancement in frequency domain: Modifying the Fourier transform of an image

Spatial Operations

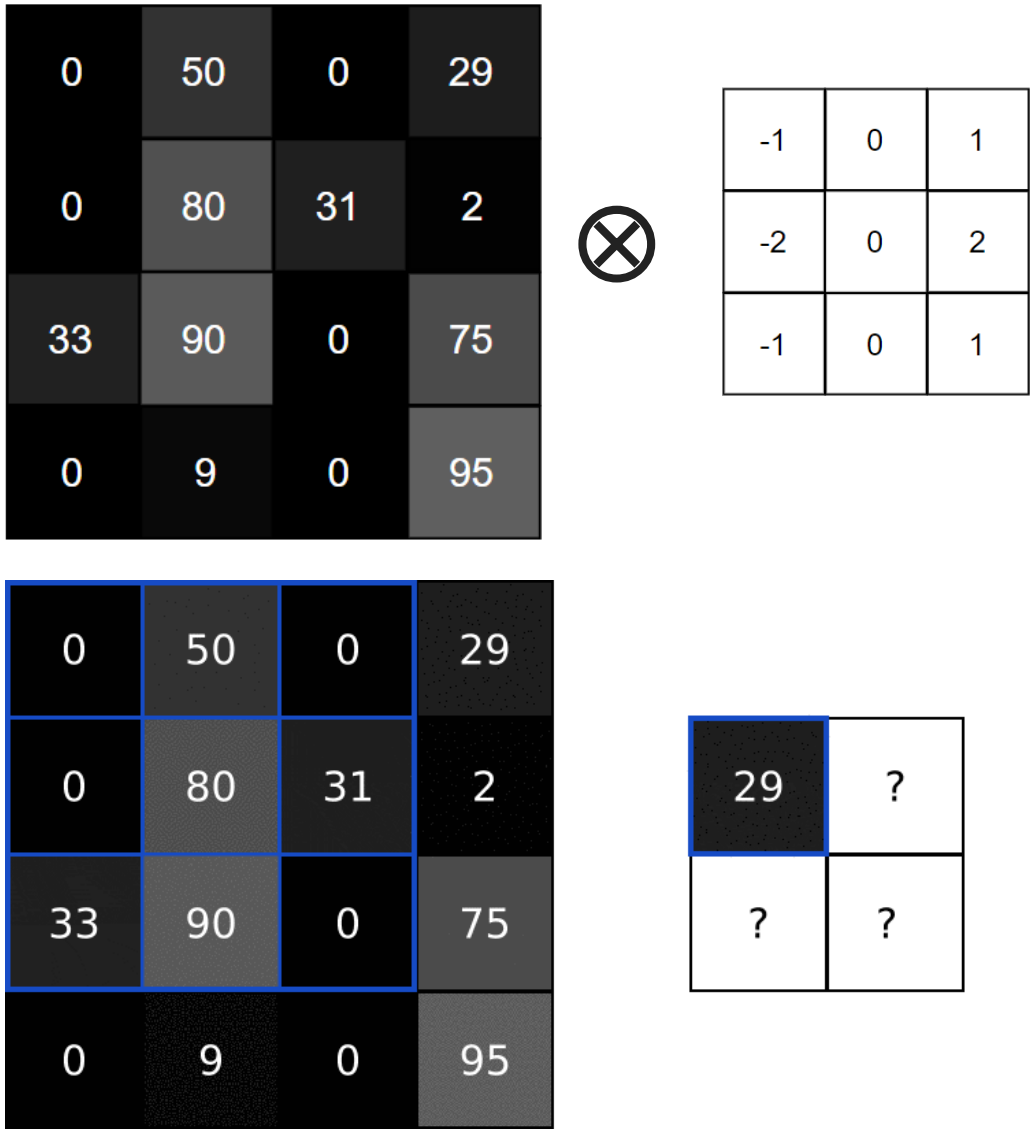
- Single-pixel operation (Intensity Transformation)
  - Negative Image, contrast stretching etc.
- Neighborhood operations
  - Averaging filter, median filtering etc.
- Geometric spatial transformations
  - Scaling, Rotation, Translations etc.





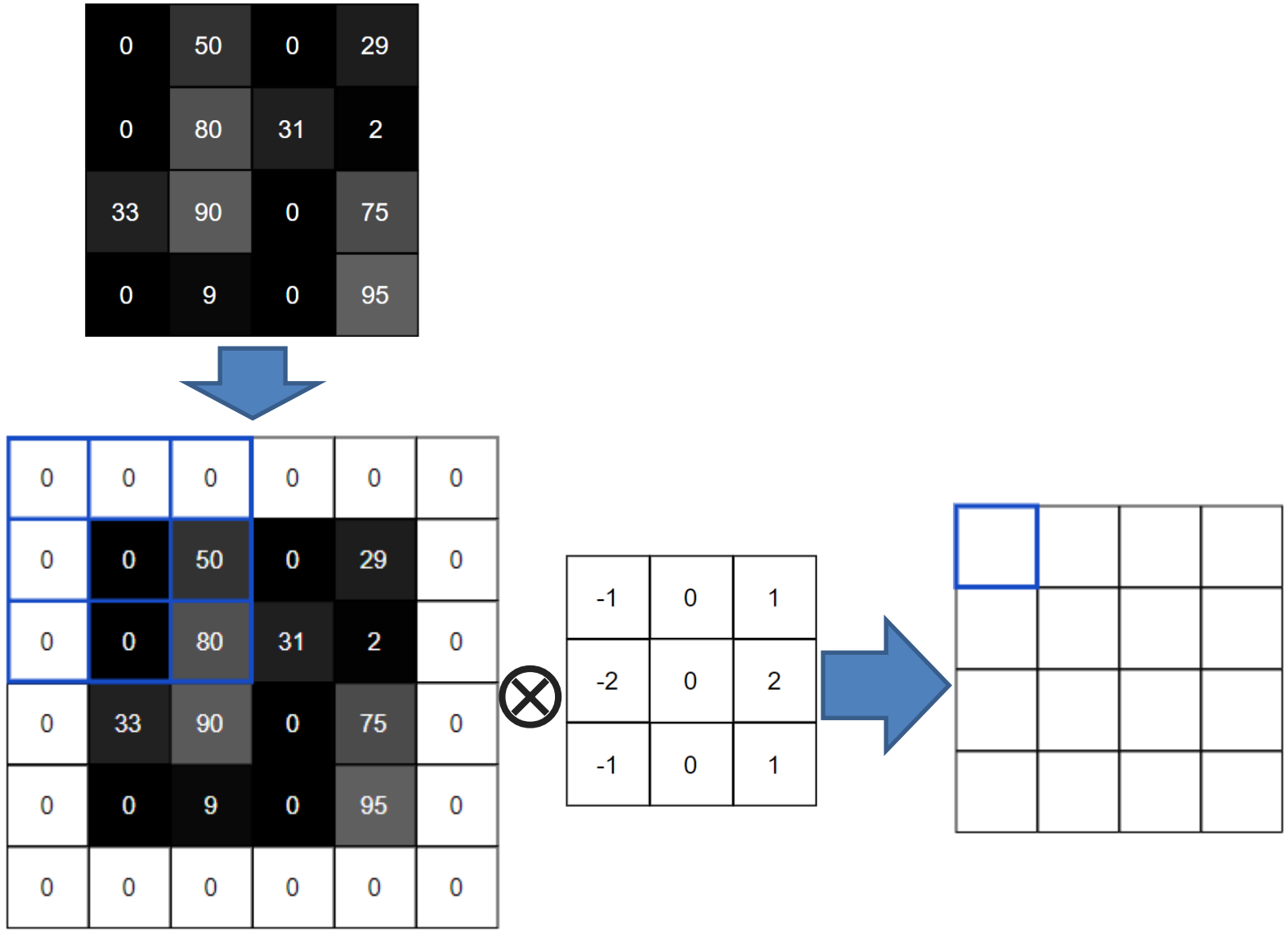
Convolution operation

- How does it work?



Convolution operation

- Padding



Quick assignment

- Read a gray image => Im, kernel k_{txt} => np.ones((t,t))/(t*t).
- Implement convolutional operator ⊗ and then convolute Out=Im⊗k.
- Then show them in plot space.

```
plt.imshow(Im)
plt.imshow(Out)

=>
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
def conv(A,k):
    kh,kw=k.shape
    h,w=A.shape
    B=np.ones((h,w))
    for i in range(0,h-kh+1):
        for j in range(0,w-kw+1):
            sA=A[i:i+kh,j:j+kw]
            B[i,j]=np.sum(k*sA)
    B=B[0:h-kh+1,0:w-kw+1]
    return B
```

Nar	Type	Size
B	Array of float64	(445, 298)
img	Array of uint8	(447, 300)
imgB	Array of uint8	(445, 298)
k	Array of float64	(3, 3)

```
img=cv2.imread('./images/Mona_Lisa.jpg',0)
k=np.ones((5,5))/25
B=conv(img,k);
imgB=np.array(B,dtype='uint8')
plt.imshow(img,cmap='gray')
plt.show()
plt.imshow(imgB,cmap='gray')
```

Quick assignment

Example

```
def conv(A,k):
    kh,kw=k.shape
    h,w=A.shape
    B=np.ones((h,w))
    for i in range(0,h-kh+1):
        for j in range(0,w-kw+1):
            sA=A[i:i+kh,j:j+kw]
            B[i,j]=np.sum(k*sA)
    B=B[0:h-kh+1,0:w-kw+1]
    return B

def conv2(A,k,b=0):
    kh,kw=k.shape
    if b>0:
        h,w=A.shape
        B=np.ones((h+kh-1,w+kw-1))
        th=int(kh/2)
        tw=int(kw/2)
        B[th:h+th,tw:w+tw]=A
        A=B
    h,w=A.shape
    C=np.ones((h,w))
    for i in range(0,h-kh+1):
        for j in range(0,w-kw+1):
            sA=A[i:i+kh,j:j+kw]
            C[i,j]=np.sum(k*sA)
    C=C[0:h-kh+1,0:w-kw+1]
    return C
```

```
img = cv2.imread('./images/Mona_Lisa.jpg')
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
k=np.ones((5,5))/25
r,g,b = cv2.split(img)
B=conv(b,k); G=conv(g,k); R=conv(r,k)
imgC=np.array(cv2.merge((R,G,B)),dtype='uint8')
plt.imshow(img)
plt.show()
plt.imshow(imgC)
```

Outline

1. Background
2. Basic intensity transformation functions
3. Histogram processing
4. Fundamentals of spatial filtering
5. Smoothing (lowpass) spatial filters
6. Sharpening (highpass) spatial filters

Negative Image

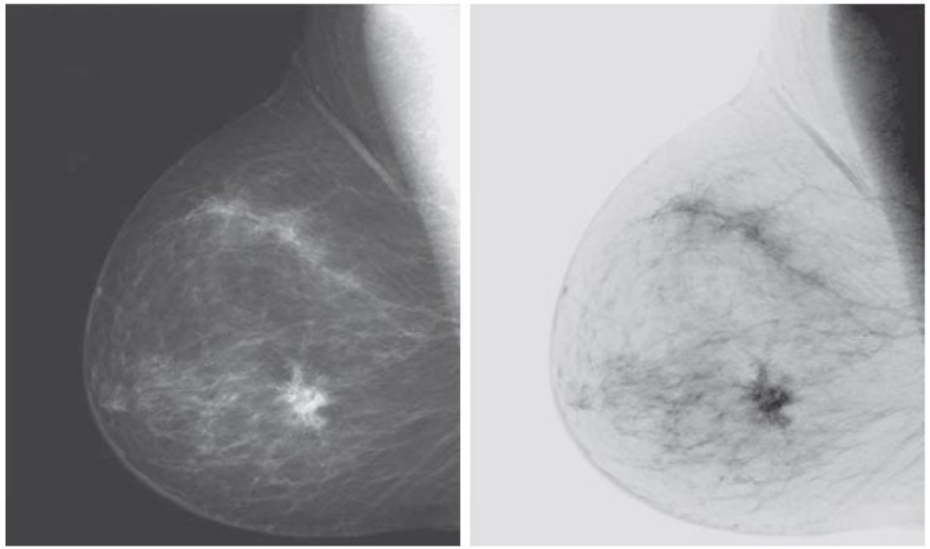
- The negative of an image with intensity levels in the range $[0, L-1]$ is obtained by using the negative transformation function

$$s = L - 1 - r$$

  - S is the output intensity value
  - L is the highest intensity levels
  - r is the input intensity value
  - Particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size

Image Negatives

Example 1:
(a) A digital mammogram
(b) Negative image



Example 2

Image Negatives

- Some basic intensity transformation functions.
- Each curve was scaled independently so that all curves would fit in the same graph.
- Our interest here is on the shapes of the curves, not on their relative values.

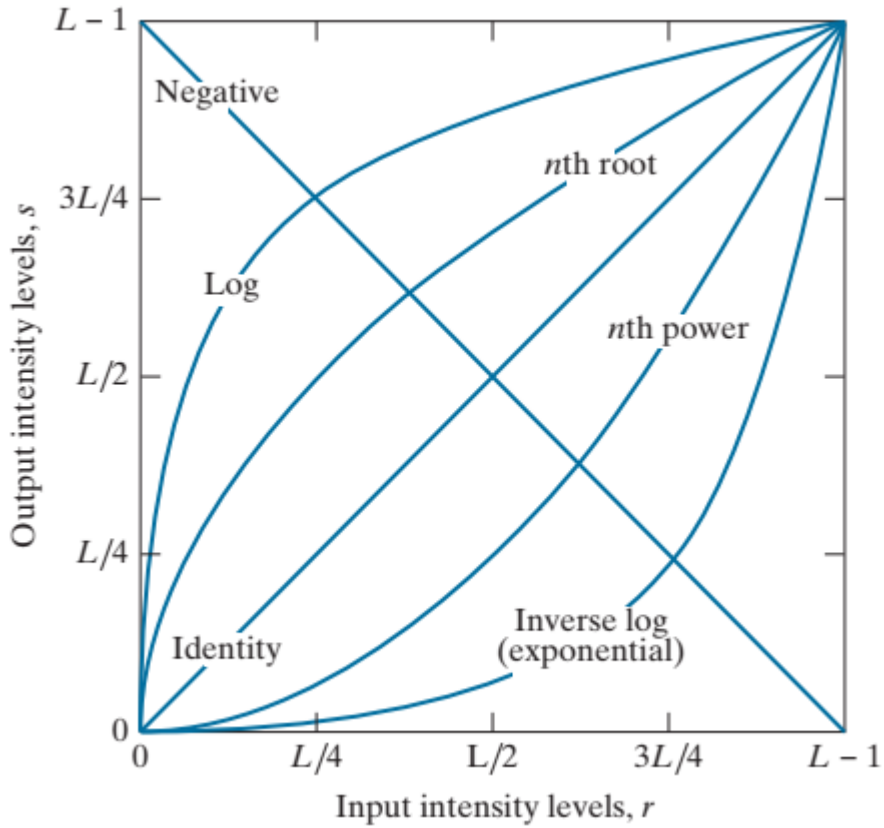
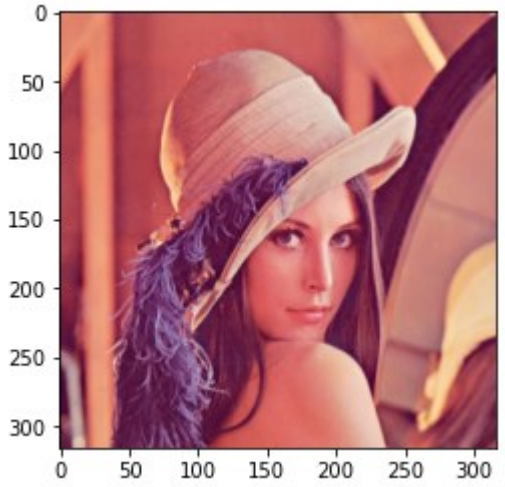
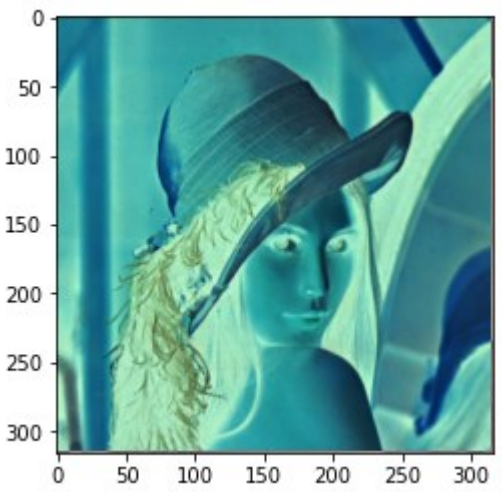


Image Negatives

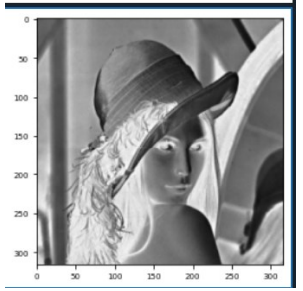
```
img_bgr = cv2.imread(pathIn+'lenna.jpg', cv2.IMREAD_COLOR)
img_neg=img_bgr
height, width, _ = img_bgr.shape
for i in range(0, height - 1):
    for j in range(0, width - 1):
        pixel = img_bgr[i, j]
        pixel[0] = 255 - pixel[0]
        pixel[1] = 255 - pixel[1]
        pixel[2] = 255 - pixel[2]
        img_neg[i, j] = pixel
# Display images
plt.imshow(cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB))
plt.show('Original Image')
plt.imshow(cv2.cvtColor(img_neg, cv2.COLOR_BGR2RGB))
plt.show('Negative transformed image')
```





```
img = cv2.imread('./Images/lenna.jpg', 0)
img_neg=256-1-img
plt.imshow(img,cmap='gray')

plt.imshow(img_neg,cmap='gray')
```

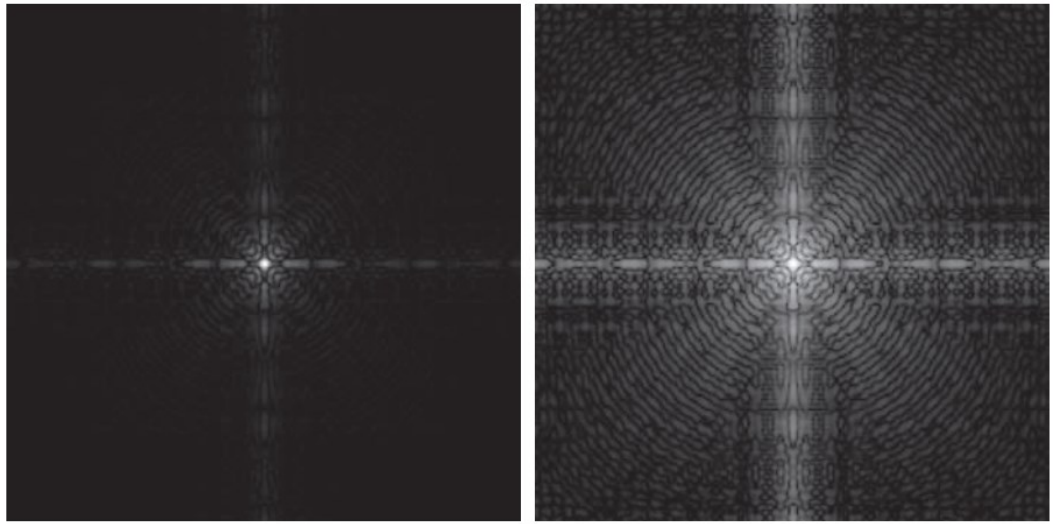


Log Transformations

$$s = c \log(1 + r)$$

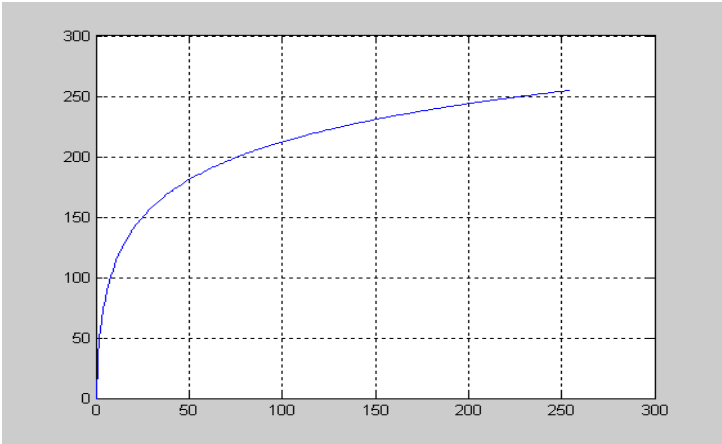
- c is constant and it is assumed that $r \geq 0$
- It maps a narrow range of low intensity values in the input into a wide range of output levels
- The opposite is true of higher values of input levels
- It expands the values of dark pixels in an image while compressing the higher level values
- It compresses the dynamic range of images with large variations in pixel values

Log Transformations



a b

FIGURE 3.5
(a) Fourier spectrum displayed as a grayscale image. (b) Result of applying the log transformation in Eq. (3-4) with $c = 1$. Both images are scaled to the range $[0, 255]$.



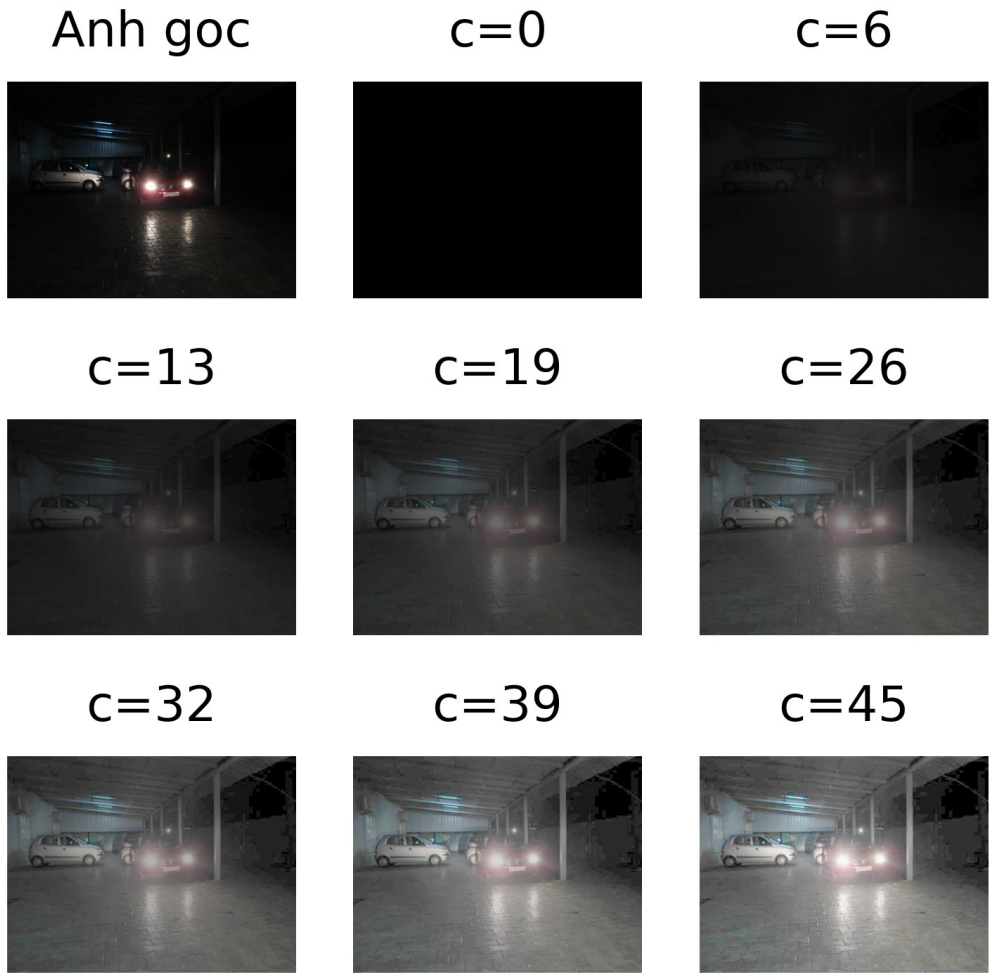
$$s = c \log(1 + r)$$

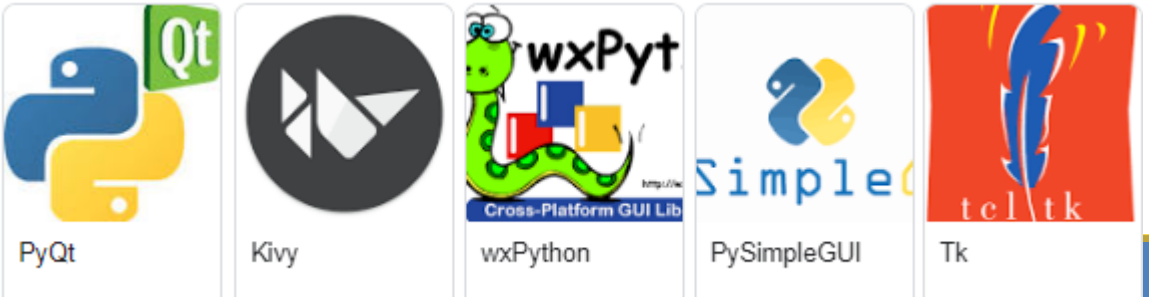
Log Transformations

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 7})
plt.figure(dpi=600)
```

- Example

```
img_bgr = cv2.imread('images/car_3.jpg')
img_bgr = cv2.cvtColor(img_bgr,cv2.COLOR_BGR2RGB)
img =np.array(img_bgr,'float')
maxV = 255 / np.log(1 + np.max(img_bgr))
vals= np.linspace(0,maxV,8,dtype=int)
plt.figure(figsize=(3,3),dpi=600)
subf=plt.subplot(3,3,1)
subf.imshow(img_bgr)
subf.set_title('Anh goc'); subf.axis('off')
for i,c in enumerate(vals):
    log_image = c * (np.log(img+1))
    log_image = np.array(log_image, dtype = 'uint8')
    subf=plt.subplot(3,3,i+2)
    subf.imshow(log_image)
    subf.set_title('c='+str(c)) ; subf.axis('off')
```





Power Law (Gamma) Transformations

$$s = cr^{\gamma}$$

- c and γ are both positive constants,

Some time, it is written as $s=c(r+\varepsilon)^{\gamma}$

- With fractional values($0<\gamma<1$) of gamma map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values ($\gamma>1$)of input levels.
- $c=\gamma=1$ means it is an identity transformations.
- Variety of devices used for image capture , printing, and display respond according to a power law.
- Process used to correct these power law response phenomena is called *gamma correction.*

Power Law (Gamma) transformations

- **Example:**

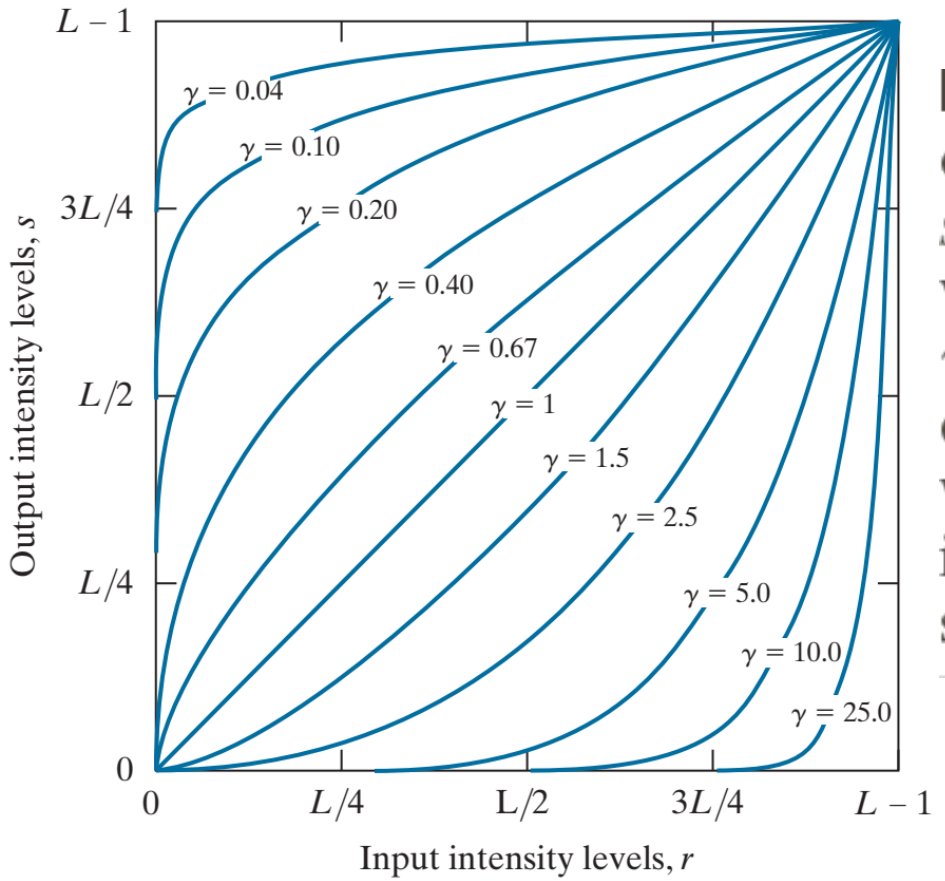
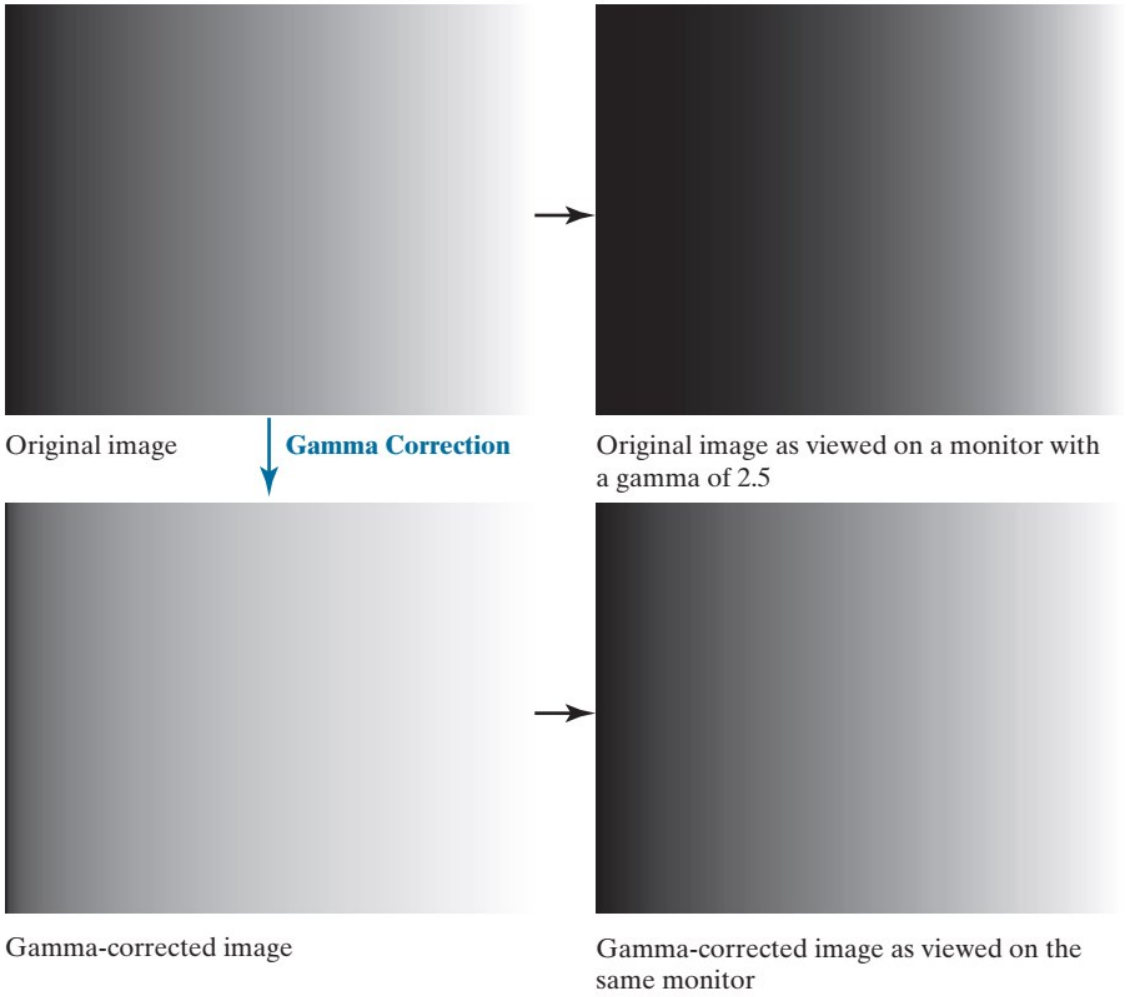


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.

Power Law (Gamma) transformations

- **Example:**



a b
c d

FIGURE 3.7
(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).

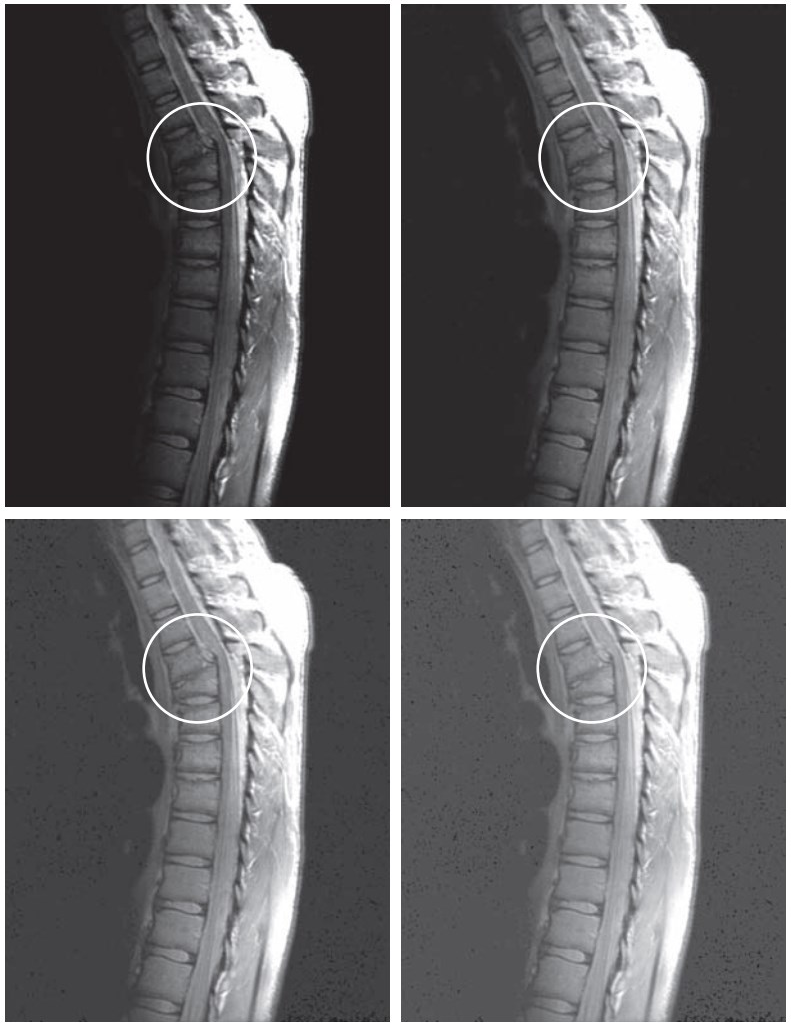
Power Law (Gamma) transformations

- Images that are not corrected properly look either bleached out or too dark.
- Varying gamma changes not only intensity, but also the ratio of red to green to blue in a color images.
- Gamma correction has become increasingly important, as the use of the digital images over internet.
- Useful for general purpose contrast manipulation.
- Apply gamma correction on CRT (Television, monitor), printers, scanners etc.
- Gamma value depends on device.

Power Law transformations

- Image is predominantly dark, => expansion of intensity levels is desirable => using values of γ smaller than 1

Example:
(a) Magnetic resonance image (MRI) of a fractured human spine (the region of the fracture is enclosed by the circle).
(b)–(d) Results of applying the Power transformation with $c = 1$ and $\gamma = 0.6, 0.4, 0.3$.



Power Law transformations

The image is washed-out appearance=> compression of intensity levels is desirable=> using values of γ greater than 1

Example:
(a) Aerial image.
(b)–(d) Results of applying the power transformation with c=1 and γ = 3.0; 4.0; 5.0

Piecewise-Linear Transformation

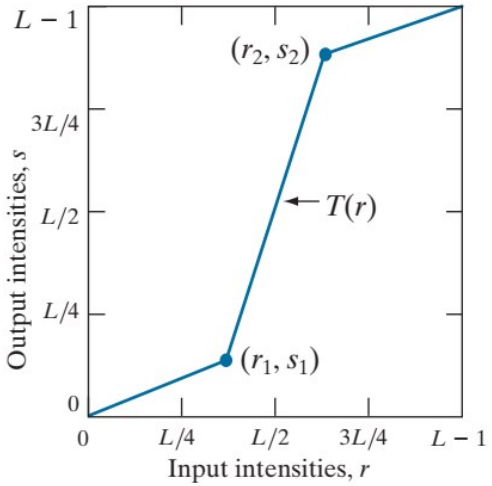
- Contrast Stretching
  - Low contrast images can result from poor illuminations.
  - Lack of dynamic range in the imaging sensor, or even the wrong setting of a lens aperture during image acquisition.
  - It expands the range of intensity levels in an image so that it spans the full intensity range of display devices.
  - Contrast stretching is obtained by setting $(r1,s1) = (r_{min}, 0)$ and $(r2,s2) = (r_{max}, L-1)$

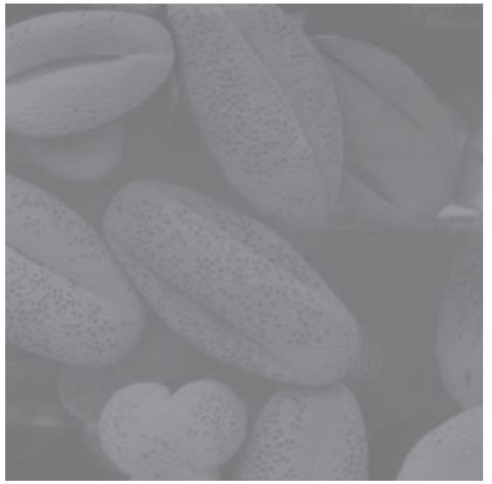
Piecewise-Linear Transformation

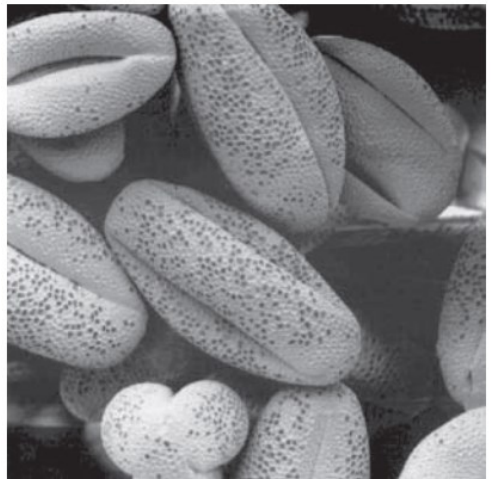
Example:

Contrast stretching.
(a) Piecewise linear transformation function.
(b) A low contrast electron microscope image of pollen, magnified 700 times.
(c) Result of contrast stretching.
(d) Result of thresholding.

(Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)







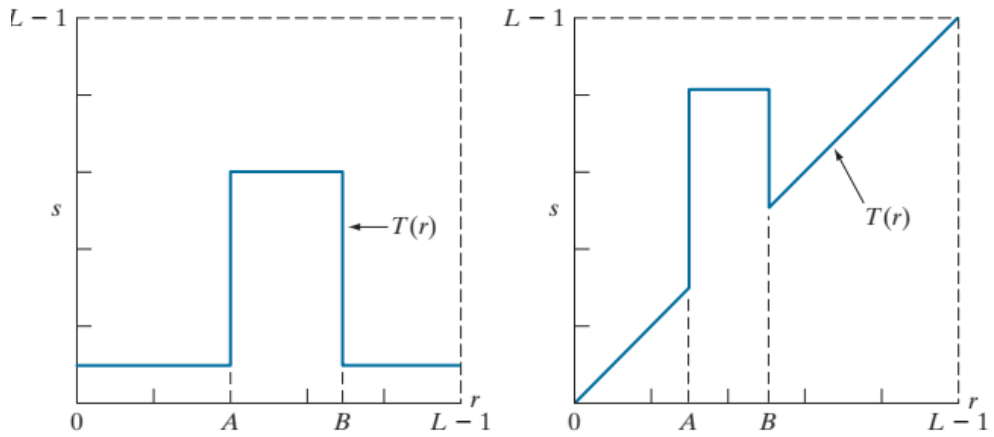
Piecewise-Linear Transformation

- **Intensity Level Slicing**
  - Highlighting specific range of intensities in an image.
  - Enhances features such as masses of water in satellite imagery and enhancing flaws in X-ray images.
  - It can be Implemented two ways:
- 1) To display only one value (say, white) in the range of interest and rests are black which produces binary image.
- 2) brightens (or darkens) the desired range of intensities but leaves all other intensity levels in the image unchanged.

Piecewise-Linear Transformation

- **Intensity Level Slicing**

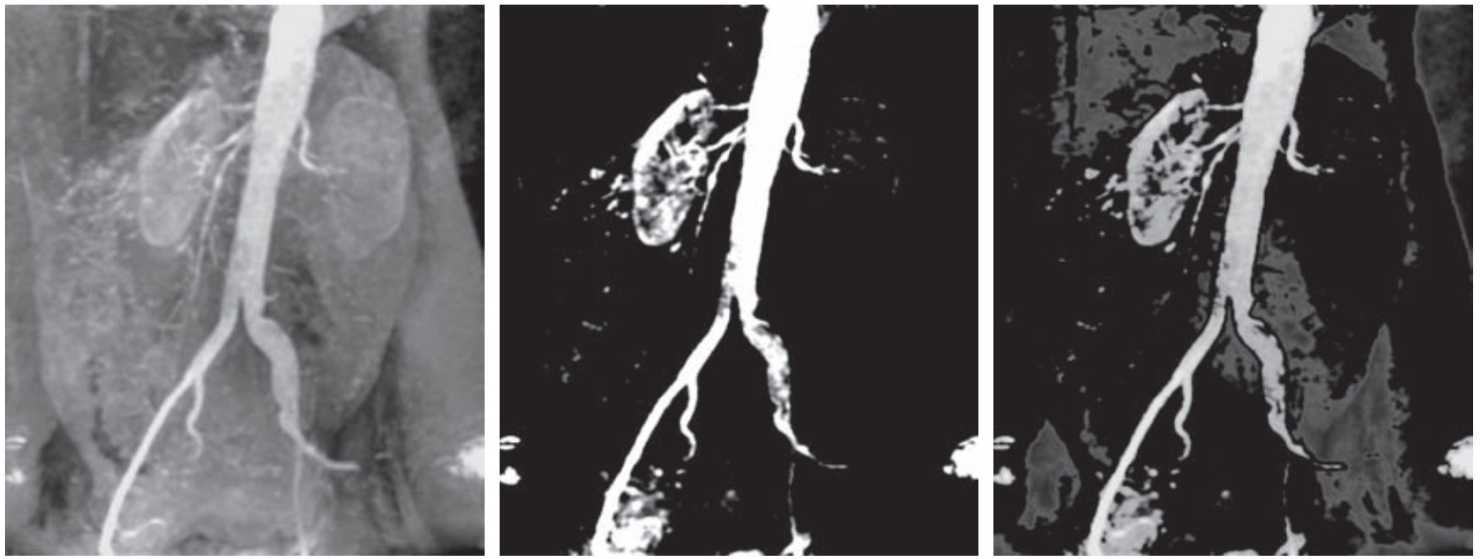
FIGURE
(a) This transformation function highlights range [*A B*] and reduces all other intensities to a lower level.
(b) This function highlights range [*A, B*] and leaves other intensities unchanged



Piecewise-Linear Transformation

Intensity Level Slicing

- Example:



a b c

FIGURE 3.12 (a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected range set near black, so that the grays in the area of the blood vessels and kidneys were preserved.

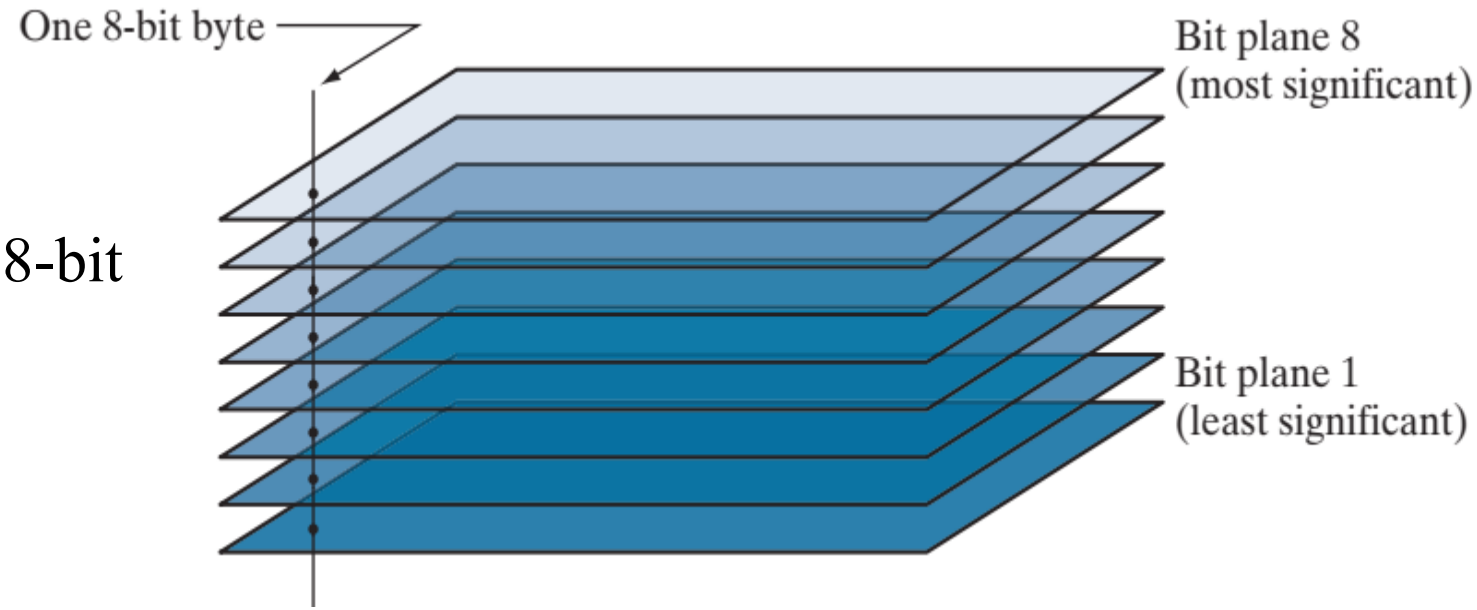
Piecewise-Linear Transformation

- Bit Plane Slicing
  - Pixels are digital numbers composed of bits.
  - 256 gray scale image is composed of 8 bits.
  - Instead of highlighting intensity level ranges, we could highlight the contribution made to total image appearance by specific bits.
  - 8-bit image may be considered as being composed of eight 1-bit planes, with plane 1 containing the lowest- order bit of all pixels in the image and plane 8 all the highest-order bits.

Piecewise-Linear Transformation

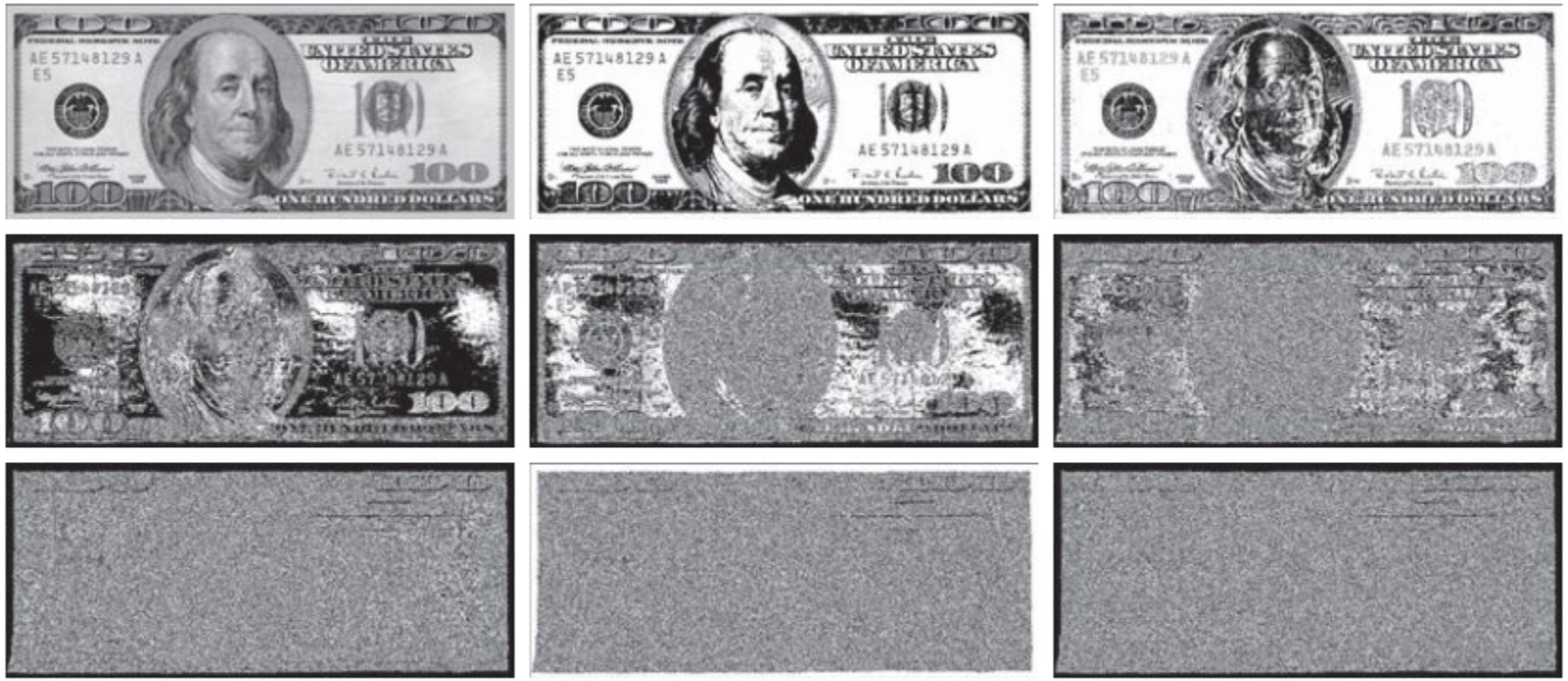
- Bit Plane Slicing

Example
Bit-planes of an 8-bit image.



Piecewise-Linear Transformation

- Bit Plane Slicing



a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 550 1192 × pixels. (b) through (i) Bit planes 8 through 1, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image..

Piecewise-Linear Transformation

- Bit Plane Slicing

a b c

FIGURE 3.15 Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

Image Enhancement with OpenCV

- Negative image

```
#include <iostream>
#include "opencv2\opencv.hpp"
#include "opencv2\imgproc.hpp"
using namespace cv;
using namespace std;
```

```
int main( int argc, char** argv )
{
    std::string img = "images\\cameraman.tif";
    Mat srcImage = imread(img);
    Mat grey; cvtColor(srcImage, grey, COLOR_BGR2GRAY);
    imshow("Original image", grey);
    double min, max;
    cv::minMaxLoc(grey, &min, &max);
    Mat dst=max-grey;
    imshow("Negative image", dst);
    waitKey(30*1000);
    destroyAllWindows();
    return 0;
}
```

Image Enhancement with OpenCV

- Log Transformation

```
std::string Simg = "images\\sample.jpg";
Mat Img = imread(Simg);
cv::imshow("Original Image",Img);
Mat Grey;
if (!Img.data) {
    return 1;
}else{
    cvtColor(Img, Grey, COLOR_BGR2GRAY);;
}
cv::Mat fg;
Grey.convertTo(fg,CV_32F);
fg = fg + 1;
cv::log(fg,fg);
cv::convertScaleAbs(fg,fg);
cv::normalize(fg,fg,0,255,cv::NORM_MINMAX);
cv::imshow("Log Image",fg);
waitKey(30*1000);
destroyAllWindows();
return 0;
```

Image Enhancement with OpenCV

- Power Law (Gamma) Transformation

```
Mat Img = imread("images\\sample.jpg",IMREAD_GRAYSCALE);
resize(Img, Img, Size(), 0.2, 0.2, INTER_NEAREST);
cv::imshow("Original Image",Img);
double gamma[5]={0.1, 0.5, 1.2, 1.5, 2.2};
double c=1;
Mat Img1;
Img.convertTo(Img1,CV_32F);
Img1=Img1/255;
Mat gamma_corrected, IResult;
for (int i= 0; i<5; i++){
    pow(Img1,gamma[i],gamma_corrected);
    gamma_corrected=255*c*gamma_corrected;
    Mat Img2;
    gamma_corrected.convertTo(Img2,CV_8UC1);
    imshow("Gamma_transformed"+std::to_string(gamma[i]),Img2);
    waitKey(50);
}
```

Image Enhancement with OpenCV

- Piecewise-Linear Transformation

```
Mat image = imread("images\\pollen.png");
Mat new_image= image.clone();
int r1 = 70, s1 = 10, r2 = 140, s2 = 200;
for(int y = 0; y < image.rows; y++){
  for(int x = 0; x < image.cols; x++){
    for(int c = 0; c < 3; c++){
     int output=compOut(image.at<Vec3b>(y,x)[c], r1, s1, r2, s2);
     new_image.at<Vec3b>(y,x)[c] = saturate_cast<uchar>(output);
    }
  }
}
imshow("Original Image",image); imshow("New Image",new_image);
```

```
int  compOut(int x, int r1, int s1, int r2, int s2)
{
    float result;
    if(0 <= x && x <= r1){
        result = s1/r1 * x;
    }else if(r1 < x && x <= r2){
        result = ((s2 - s1)/(r2 - r1)) * (x - r1) + s1;
    }else if(r2 < x && x <= 255){
        result = ((255 - s2)/(255 - r2)) * (x - r2) + s2;
    }
    return (int)result;
}
```

Outline

1. Background
2. Basic intensity transformation functions
3. Histogram processing
4. Fundamentals of spatial filtering
5. Smoothing (lowpass) spatial filters
6. Sharpening (highpass) spatial filters

Histogram Processing

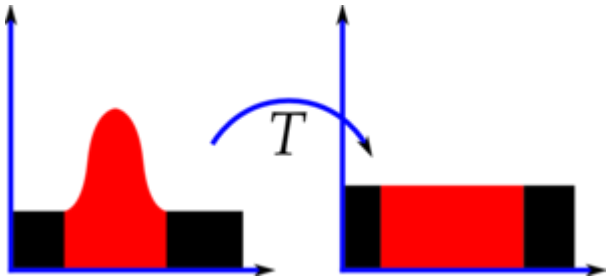
- Histogram of a digital image with intensity levels in the range [0, L-1] is a discrete function $h(r_k) = n_k$, where r_k is the k^{th} intensity value and n_k is the number of pixels in the image with intensity r_k

=> Normalized histogram: $p(r_k) = \frac{n_k}{MN}$ for k = 0,1,2……. L-1

- Histogram manipulation can be used for image enhancement.
- Histogram is quite useful in other image processing applications, e.g. image compression and segmentation.

Histogram Equalization

- It is a method in image processing of contrast adjustment using the image's histogram.
- Histogram equalization is one of the best methods for image enhancement.



Histograms of an image before and after equalization.

Histogram Equalization

- Intensity mapping form

$$s = \boldsymbol{T}(\boldsymbol{r}) \text{ for } 0 \le r \le L-1$$

Conditions:

a) T(r) is a monotonically increasing function in interval [0, L-1]

b) $0 \le T(r) \le L-1 \ \ for \ \ 0 \le r \le L-1$

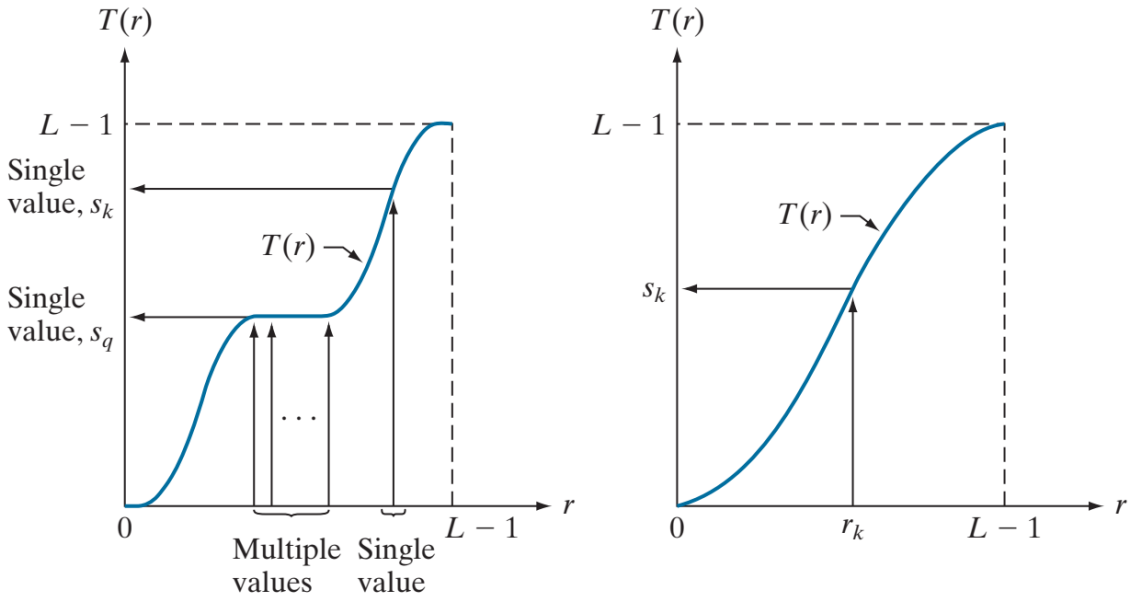
In some formulations, we use the inverse in which case

$r = T^{-1}(s),\ 0 \le s \le 1$

Condition (a) change to (a'):

(a') that T(r) is a strictly monotonically increasing function in the interval [0, L-1]

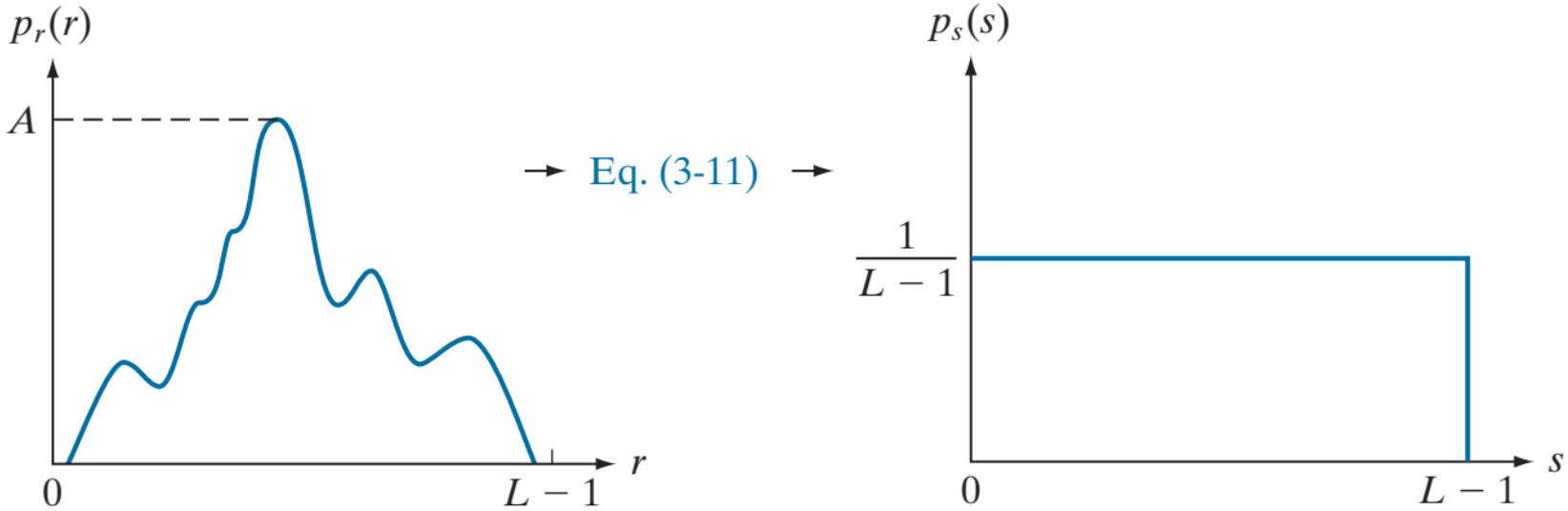
Histogram Processing



a b

FIGURE 3.17 (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

Histogram Processing



a b

(a) An arbitrary PDF. (b) Result of applying transformation to the input PDF. The resulting PDF is always uniform, independently of the shape of the input

Histogram Equalization

- Intensity levels in an image may be viewed as random variables in the interval [0,L-1]
- Fundamental descriptor of a random variable is its probability density function (PDF)
- Let $p_r(r)$ and $p_s(s)$ denote the PDFs of r and s

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

$$= p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right|$$

$$= \frac{1}{L-1} \qquad 0 \le s \le L-1$$

$$p_r(r) = \begin{cases} \dfrac{2r}{(L-1)^2} & \text{for } 0 \le r \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = (L-1)p_r(r)$$

Histogram Processing

- Example

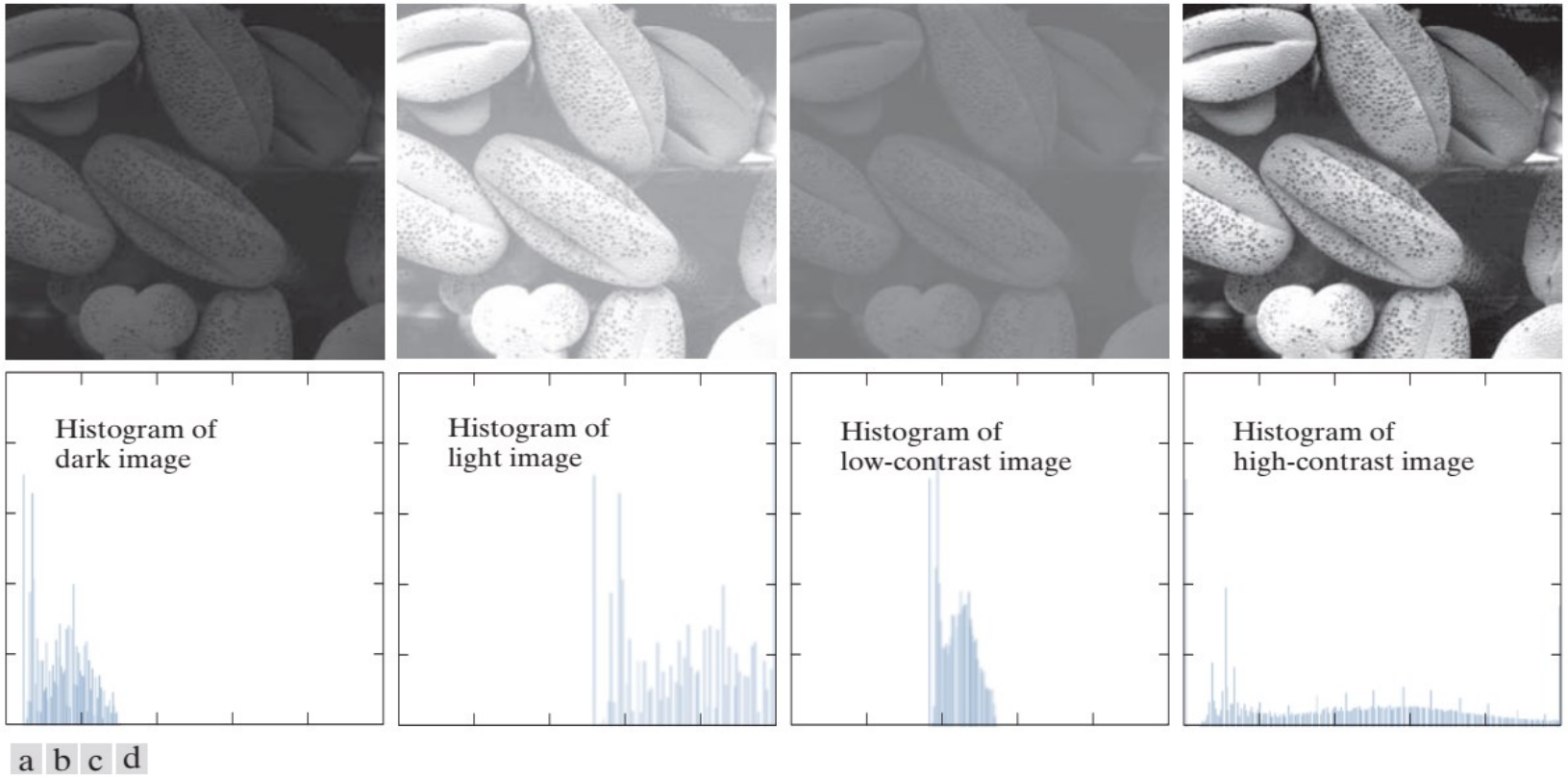


FIGURE 3.16 Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; (d) high contrast. The horizontal axis of the histograms are values of r_k and the vertical axis are values of $p(r_k)$.

Histogram Equalization

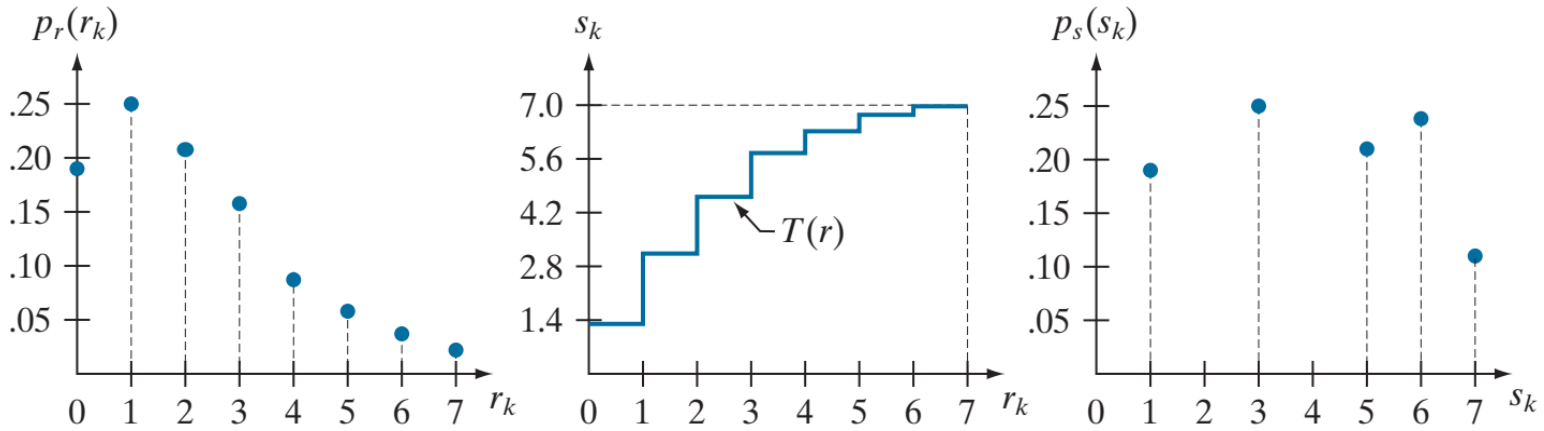
TABLE 3.1
Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Histogram Equalization

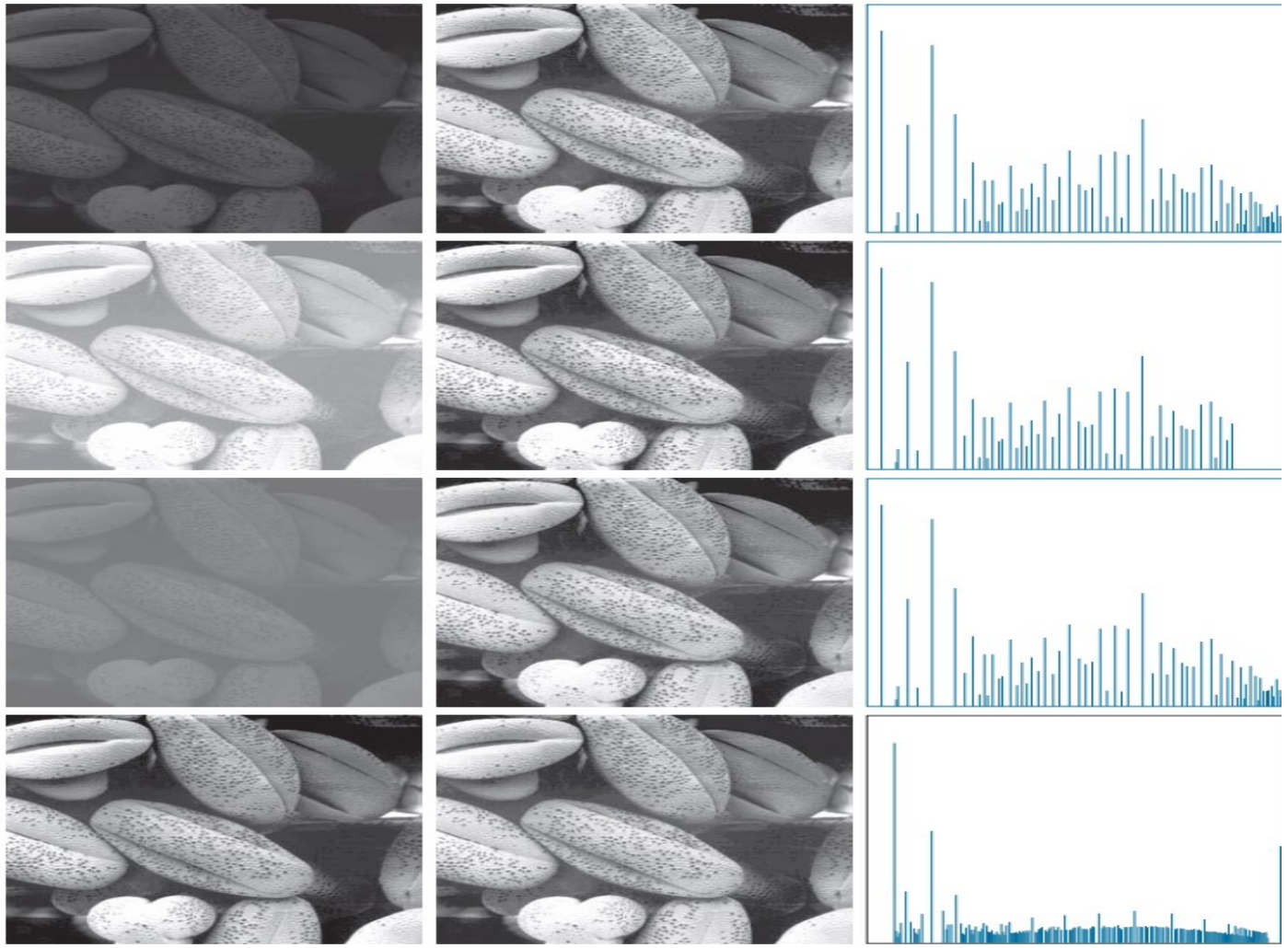
a b c

FIGURE 3.19
Histogram equalization.
(a) Original histogram.
(b) Transformation function.
(c) Equalized histogram.



Histogram Equalization

FIGURE 3.20
- Left column: Original Images
- Center column: histogram-equalized images.
- Right column: histograms of the images in the center column



Histogram Equalization

- Transformation functions

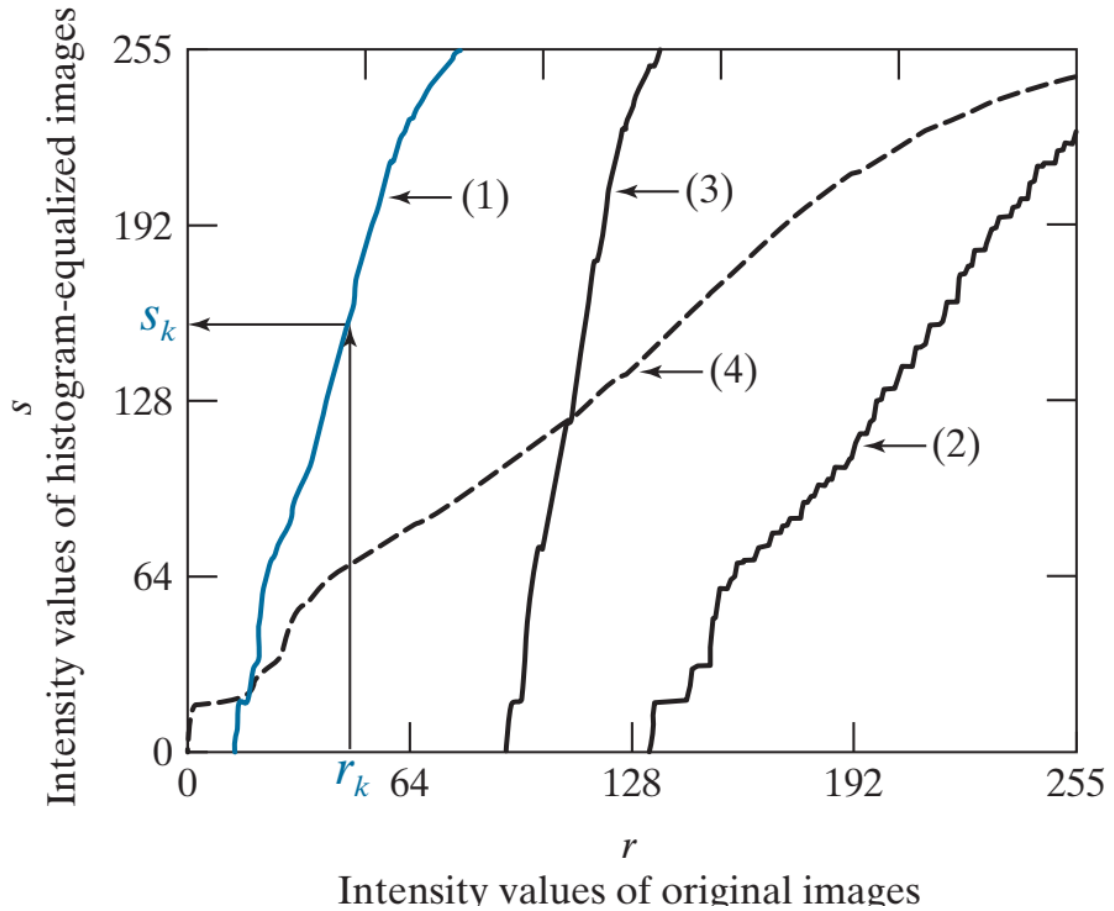


FIGURE 3.21
Transformation functions for histogram equalization.
Transformations (1) through (4) were obtained using Eq. (3-15) and the histograms of the images on the left column of Fig. 20.
Mapping of one intensity value r_k in image 1 to its corresponding value s_k

Histogram Matching

- Histogram equalization automatically determines a transformation function produce uniform histogram
- When automatic enhancement is desired, equalization is a good approach
- There are some applications in which attempting to base enhancement on a uniform histogram is not the best approach
- In particular, it is useful sometimes to be able to specify the shape of the histogram that wish the processed image to have.
- Method used to generate a processed image that has a specified histogram is called *histogram matching* or *specification*

Histogram Matching (Specification)

- Histogram specification procedure:

  1)Compute the histogram $p_r(r)$ of the given image, and use it to find the histogram equalization transformation in equation $s_k=\mathrm{T}(r_k)=(L-1)\sum_{j=0}^{k}\frac{n_j}{MN}$, $\mathrm{k}=0,1,2,..,L-1$ and round the resulting values to the integer range [0, L-1]

  2) Compute all values of the transformation function G using same equation $G(z_q)=(L-1)\sum_{i=0}^{q}p_z(r_i)$, $q=0,1,2,..,L-1$ and round values of G

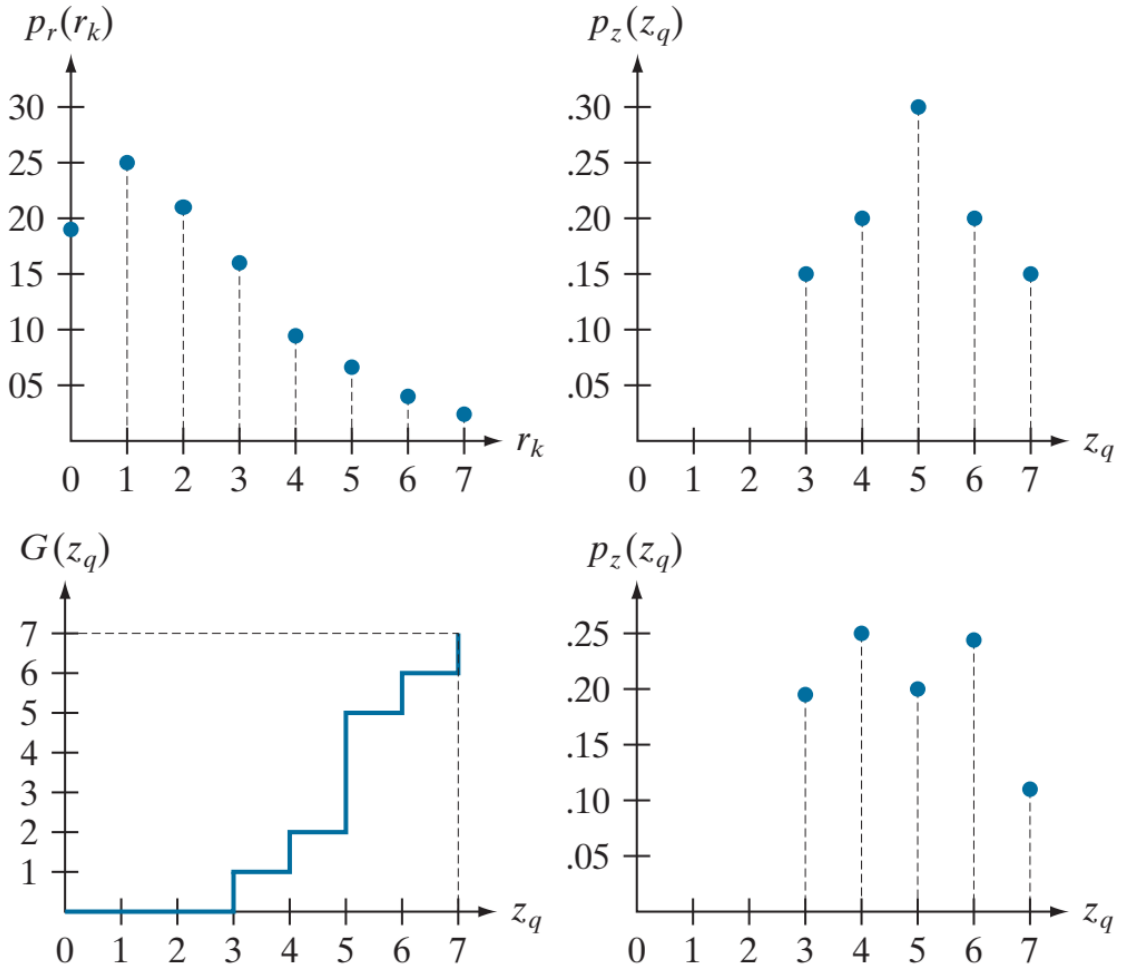
Histogram Matching (Specification)

- Histogram Specification Procedure:

  3) For every value of s_k, k = 0,1,…,L-1, use the stored values of G to find the corresponding value of z_q so that $G(z_q)$ is closet to s_k and store these mappings from s to z.

  4) Form the histogram-specified image by first histogram- equalizing the input image and then mapping every equalized pixel value, s_k , of this image to the corresponding value z_q in the histogram-specified image using the mappings found in step 3.

Histogram Matching



a b
c d

FIGURE 3.22
(a) Histogram of a 3-bit image. (b) Specified histogram. (c) Transformation function obtained from the specified histogram. (d) Result of performing histogram specification. Compare (b) and (d).

Histogram Matching

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

TABLE 3.3
All possible values of the transformation function G scaled, rounded, and ordered with respect to z.

r_k	S_k	$G(Z_q)$
0	1	0
1	3	0
2	5	0
3	6	1
4	6	2
5	7	5
6	7	6
7	7	7

s_k	$\rightarrow$	z_q
1	$\rightarrow$	3
3	$\rightarrow$	4
5	$\rightarrow$	5
6	$\rightarrow$	6
7	$\rightarrow$	7

TABLE 3.4
Mappings of all the values of s_k into corresponding values of z_q.

Histogram Matching

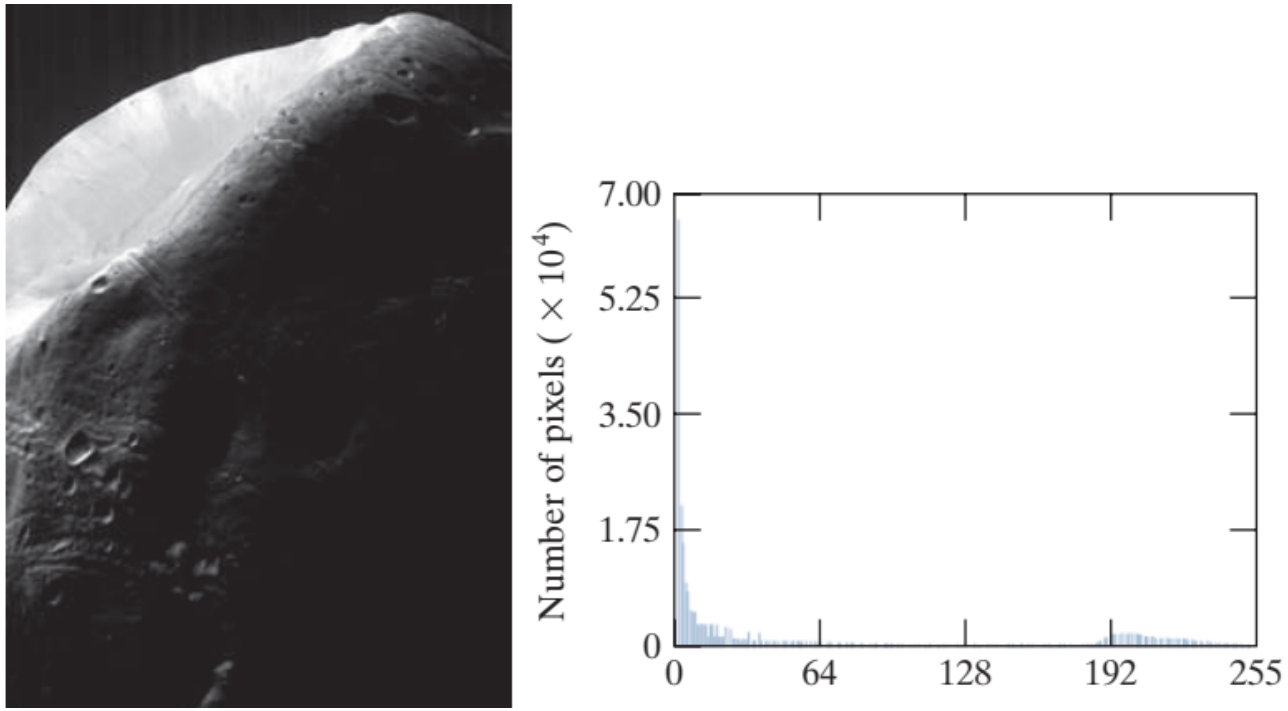
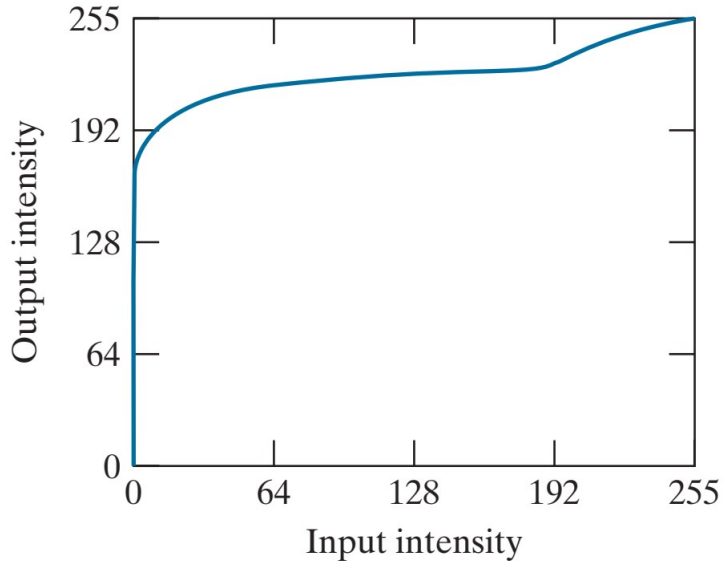


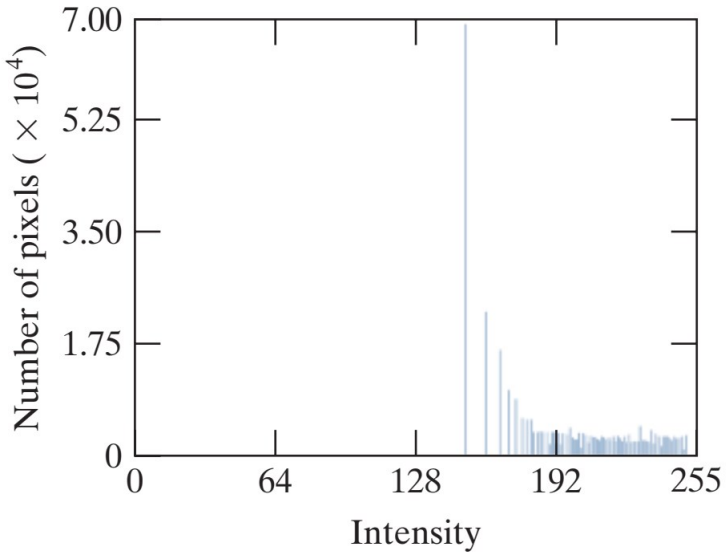
FIGURE 3.23
(a) An image,
(b) its histogram.

Histogram Matching

a b
c

FIGURE 3.24
(a) Histogram equalization transformation obtained using the histogram in Fig. 3.23(b). (b) Histogram equalized image. (c) Histogram of equalized image.

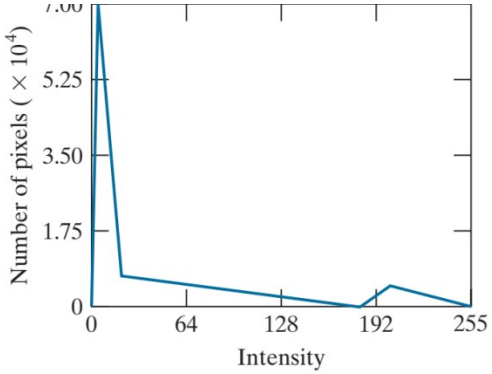


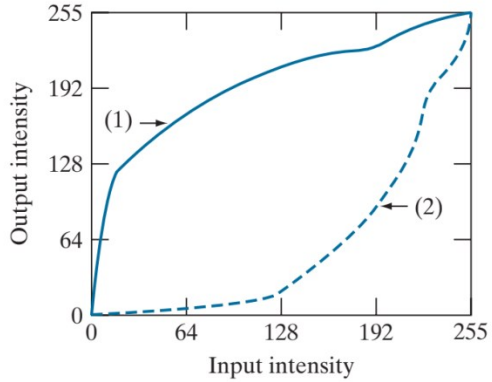


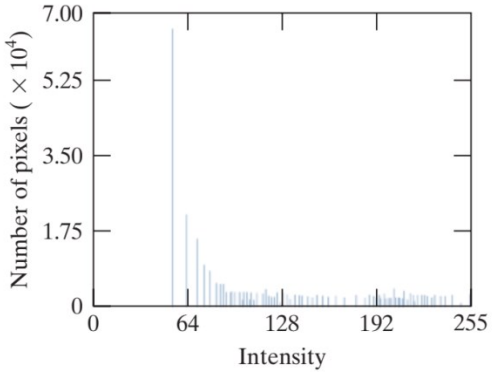
Histogram Matching

a c
b
d

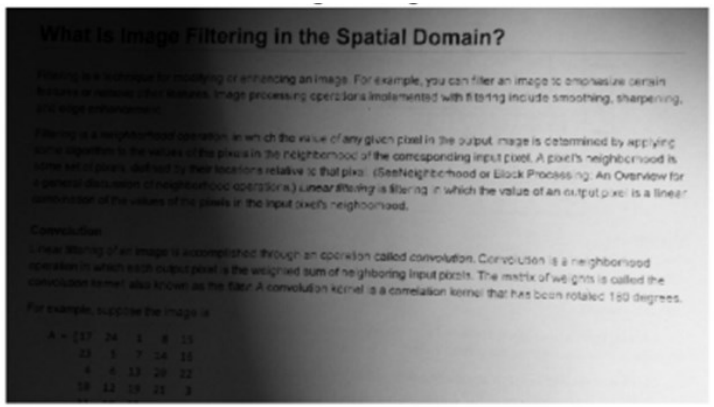
FIGURE 3.25
Histogram specification.
(a) Specified histogram.
(b) Transformation $G(z_q)$, labeled (1), and $G^{-1}(s_k)$, labeled (2).
(c) Result of histogram specification.
(d) Histogram of image (c).

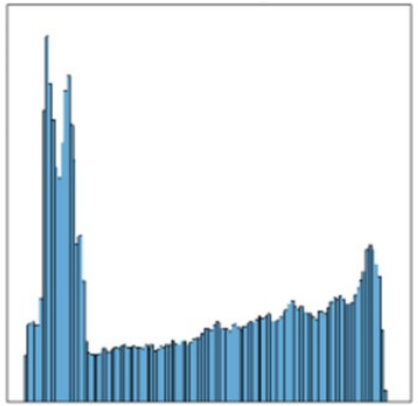


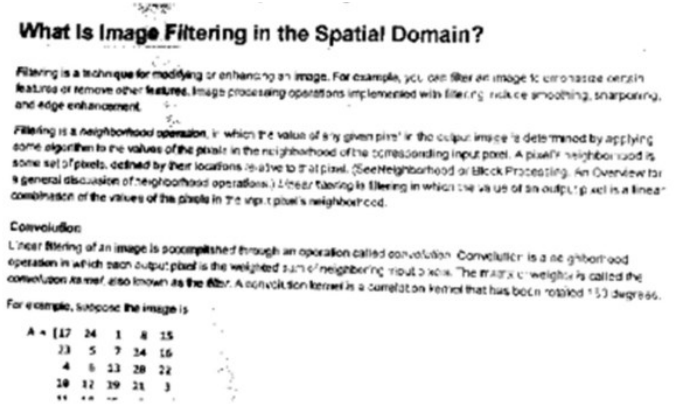




Local Histogram Processing







Local Histogram Processing

- Histogram Processing methods discussed in the previous two sections are *Global,* in the sense that pixels are modified by a transformation function based on the intensity distribution of an entire image.
- There are some cases in which it is necessary to enhance detail over small areas in an image.
- This procedure is to define a neighborhood and move its center pixel to pixel.
- At each location, the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained.
- Map the intensity of the pixel centered in the neighborhood.
- Center of the neighborhood region is then moved to an adjacent pixel location and the procedure is repeated.

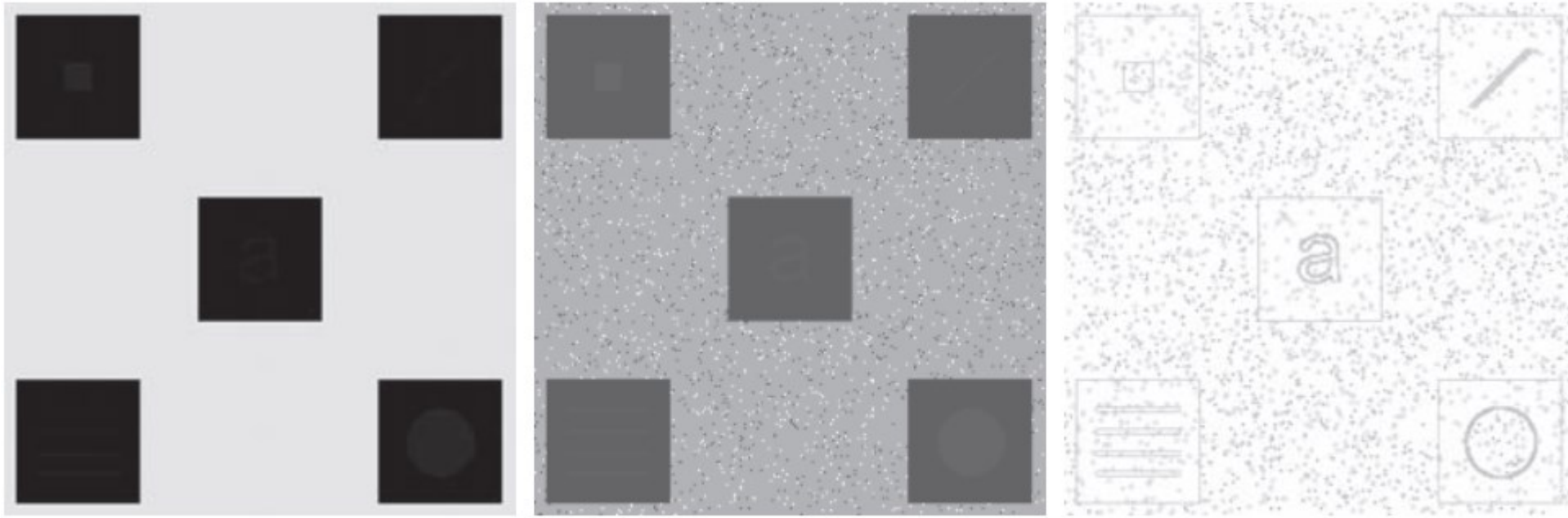
Local Histogram Processing

- This approach has obvious advantages over repeatedly computing the histogram of all pixels in the neighborhood region each time the region is moved one pixel location.

- Another approach used sometimes to reduce computation is to utilize non overlapping regions, but this method usually produces an undesirable *"blocky"* effect.

Local Histogram Processing

- Example



Outline

1. Background
2. Basic intensity transformation functions
3. Histogram processing
4. Fundamentals of spatial filtering
5. Smoothing (lowpass) spatial filters
6. Sharpening (highpass) spatial filters

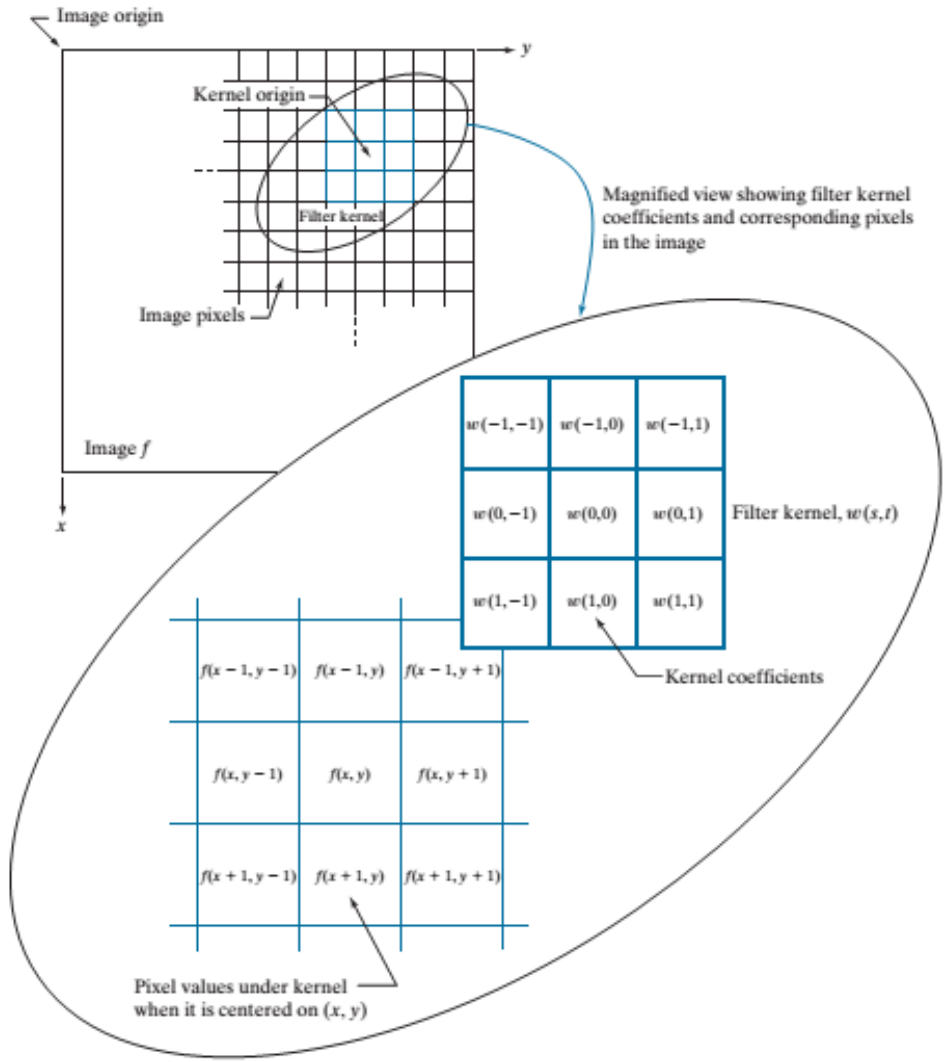
Spatial Filtering

- A linear spatial filter performs a sum-of-products operation between an image f and a filter kernel, w
- Also refer tp spatial filter kernel are masks, templates, and windows.
- It consists of (1) a neighborhood (small window), and (2) a predefined operation that is performed on the image pixels encompassed by the neighborhood
- Filtering creates a new pixel with coordinates equal to the center of the neighborhood.
- If operation is linear, then filter is called a *linear spatial filter* otherwise *nonlinear.*

Mechanics of Spatial Filtering

• The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center.

•Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.



Spatial Correlation & Convolution

- Correlation is the process of moving a filter mask over the image and computing the sum of the products at each location.

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots + w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1)$$

- Convolution process is same except that the filter is first rotated by 180 degree.

➢ If the values of a kernel are symmetric about its center, correlation and convolution yield the same result.

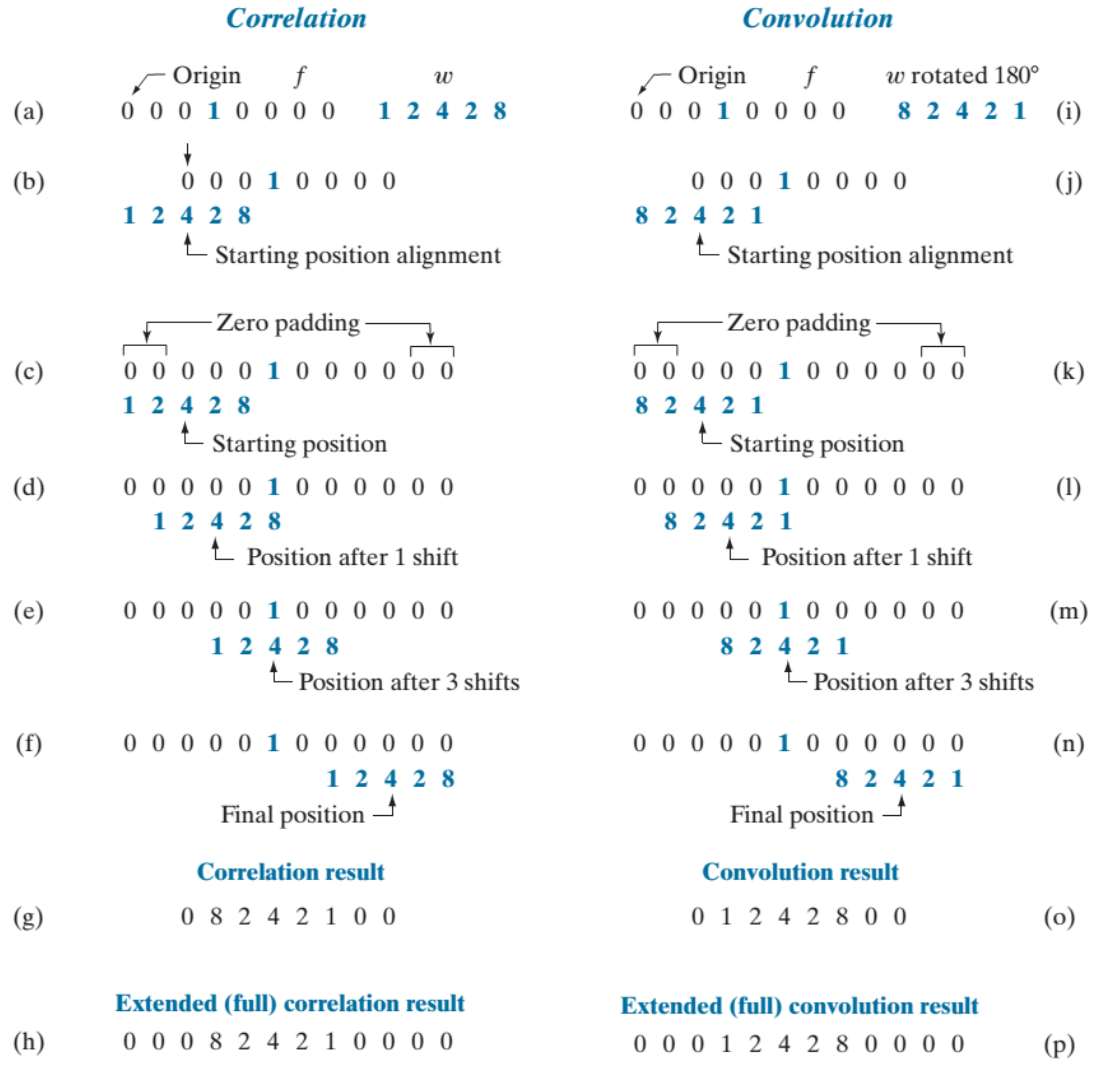
Spatial Correlation & Convolution

• Illustration of 1-D correlation and convolution of a kernel, w, with a function f consisting of a discrete unit impulse.

$$g(x) = \sum_{s=-a}^{a} w(s) f(x+s)$$

•Note that correlation and convolution are functions of the variable x, which acts to displace one function with respect to the other.

• For the extended correlation and convolution results, the starting configuration places the rightmost element of the kernel to be coincident with the origin of f. Additional padding must be used.

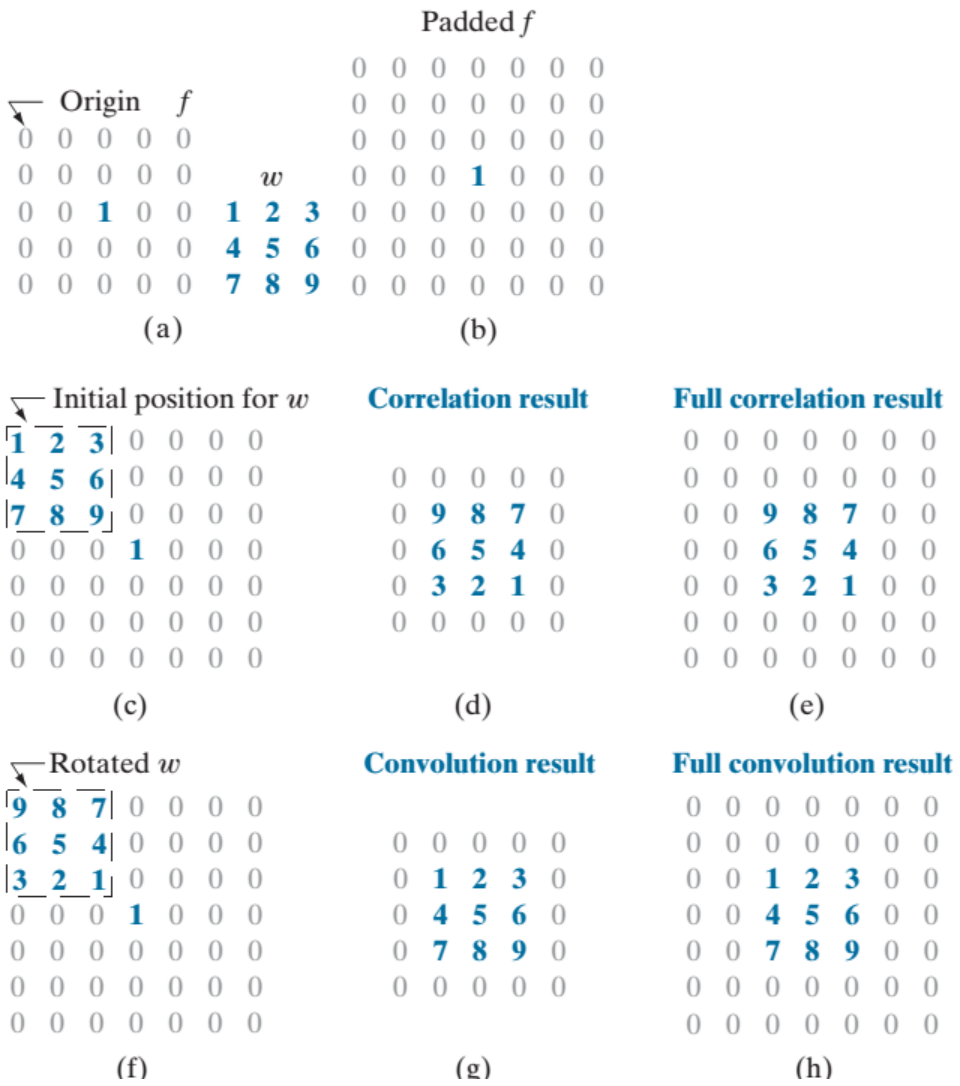


Spatial Correlation & Convolution

- Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse.

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

- The 0's are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of x and y. As these variable change, they displace one function with respect to the other. See the discussion of Eqs. (3-36) and (3-37) regarding full correlation and convolution.

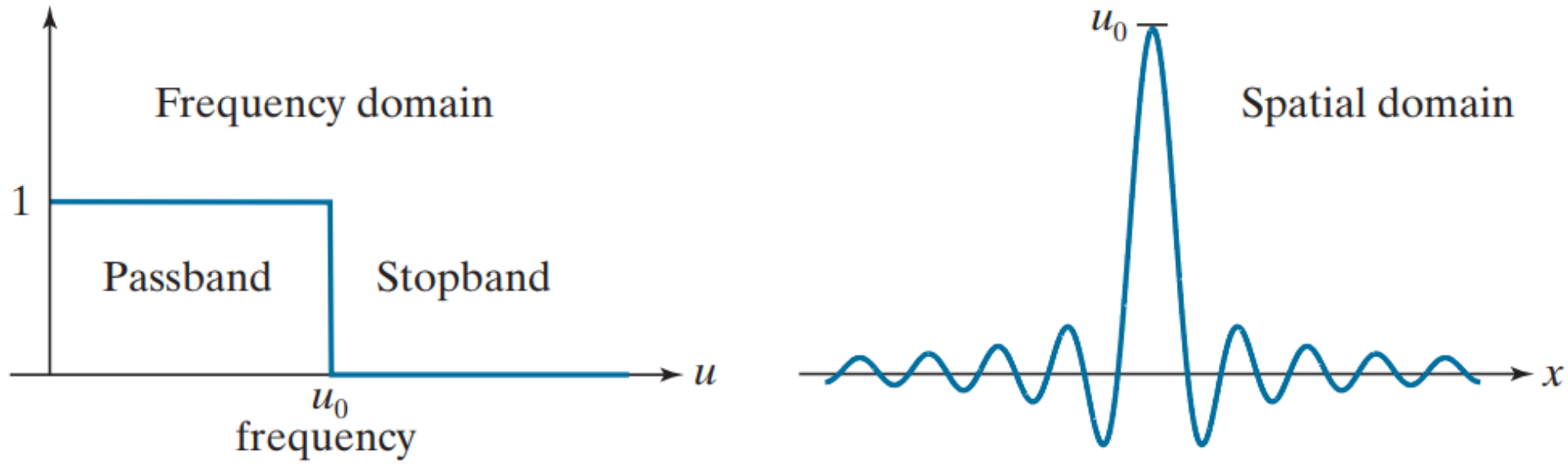


Outline

1. Background
2. Basic intensity transformation functions
3. Histogram processing
4. Fundamentals of spatial filtering
5. Smoothing (lowpass) spatial filters
6. Sharpening (highpass) spatial filters

Smoothing Spatial Filters

- Fig.a shows a frequency-domain filter function. Fig.b shows the spatial filter kernel corresponding to the frequency domain filter transfer function in Fig.a.
- It called a **lowpass** filter transfer function.



Smoothing Spatial Filters

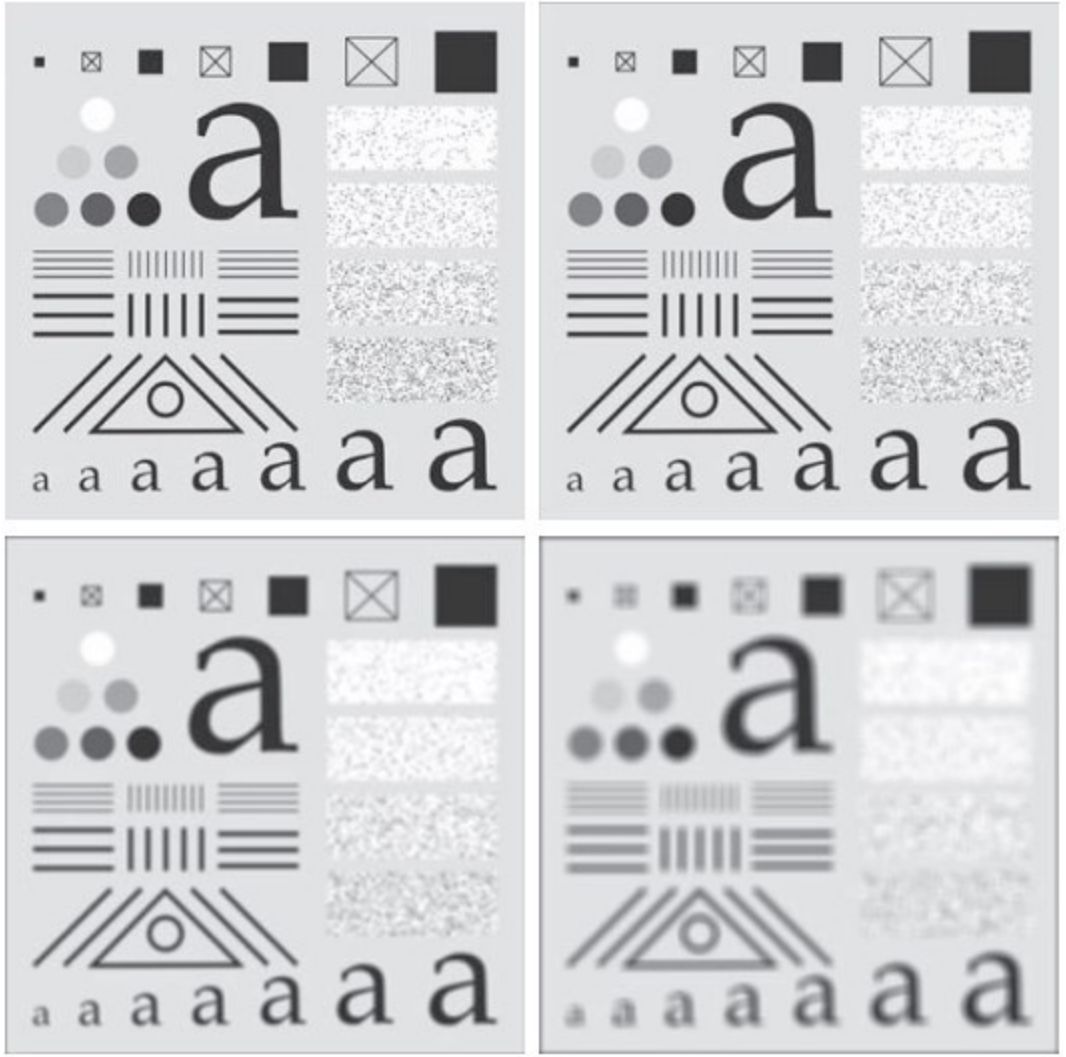
- Smoothing also called *averaging* or *Lowpass filter.*
- By replacing the value of every pixel in an image by the average of the intensity levels in the neighborhood defined by the filter mask.
- Reduced “sharp” transition in intensities.
- Random noise typically consists of sharp transition.
- Edges also characterized by sharp intensity transitions, so averaging filters have the undesirable side effect that they blur edges.
- If all coefficients are equal in filter than it is also called *a box filter*.

Smoothing Spatial Filters

- Smoothing filter is used for blurring and noise reduction in the image.
  - Blurring is pre-processing steps for removal of small details
  - Noise reduction is accomplished by blurring.

Smoothing Spatial Filters

(a) Test pattern of size 1024 ×1024 pixels.
(b)-(d) Results of lowpass filtering with box kernels of sizes 3×3, 11×11, and 21×21.



Box filter kernels

- The name “box kernel” comes from a constant kernel resembling a box when viewed in 3-D
- Example 3×3 box filter

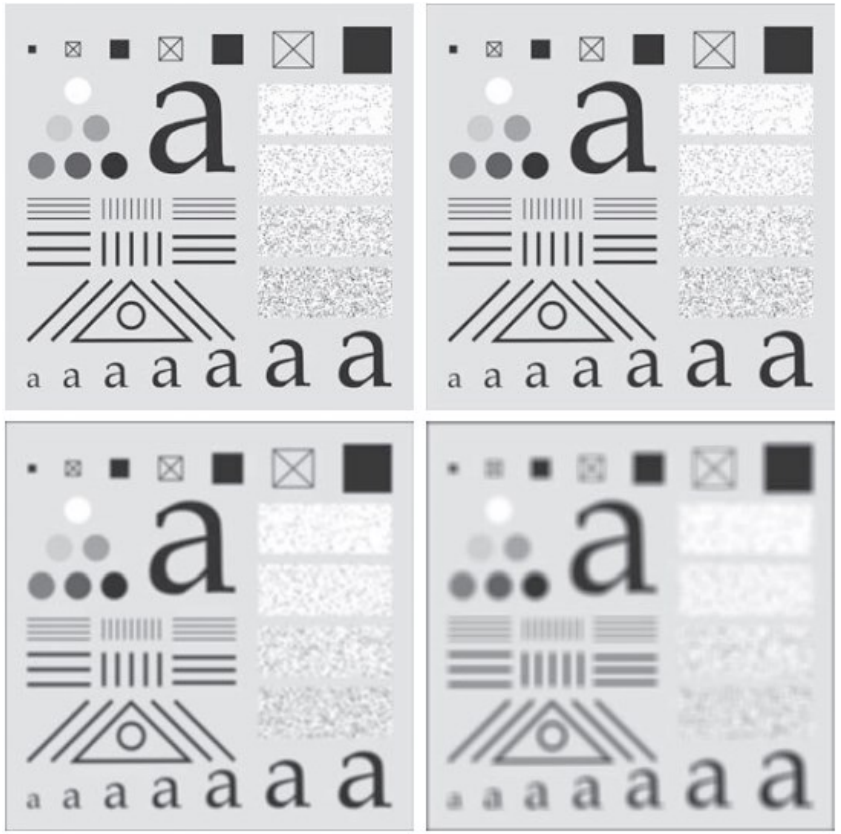
$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Box filter kernels

Lowpass filtering with a box kernel

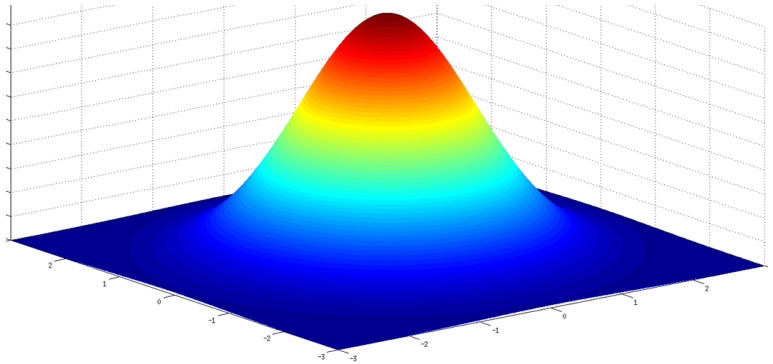
- Example shows a test pattern image of size 1024×1024 pixels. Figures (b)-(d) are the results obtained using box filters of size m×m with m= 3, 11 and 21.

➢ m = 3, we note a slight overall blurring of the image, with the image features whose sizes are comparable to the size of the kernel being affected significantly more.



Gaussian filter kernels

- Phân phối Gaussian trong 2D và 3D



- Gaussian kernels of the form

$$G(r) = Ke^{-\frac{r^2}{2\sigma^2}}$$

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$

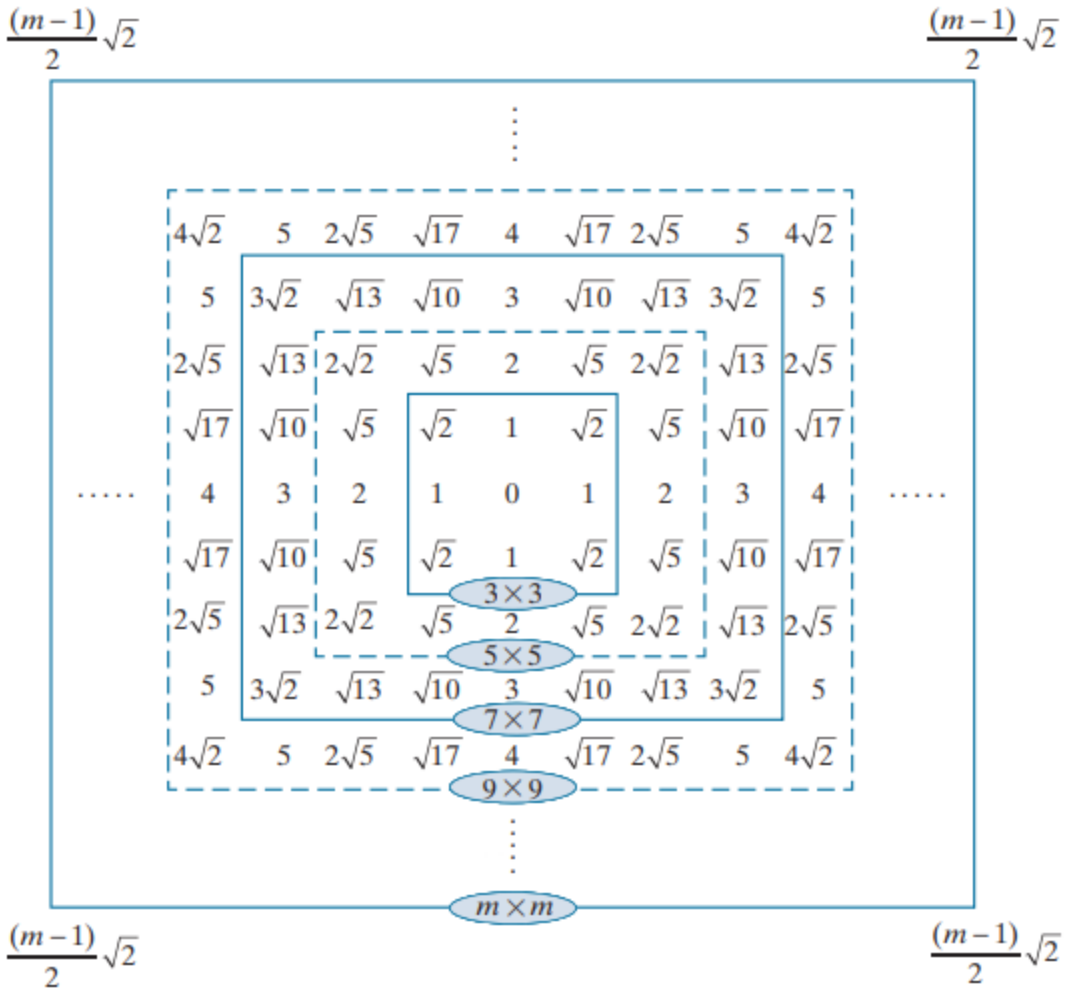
Variables r, s, t are real (typically discrete) numbers
K is a constance

Gaussian filter kernels

- Distances from the center for various sizes of square kernels.

- Note in particular that the distance squared to the corner points for a kernel of size m×m is

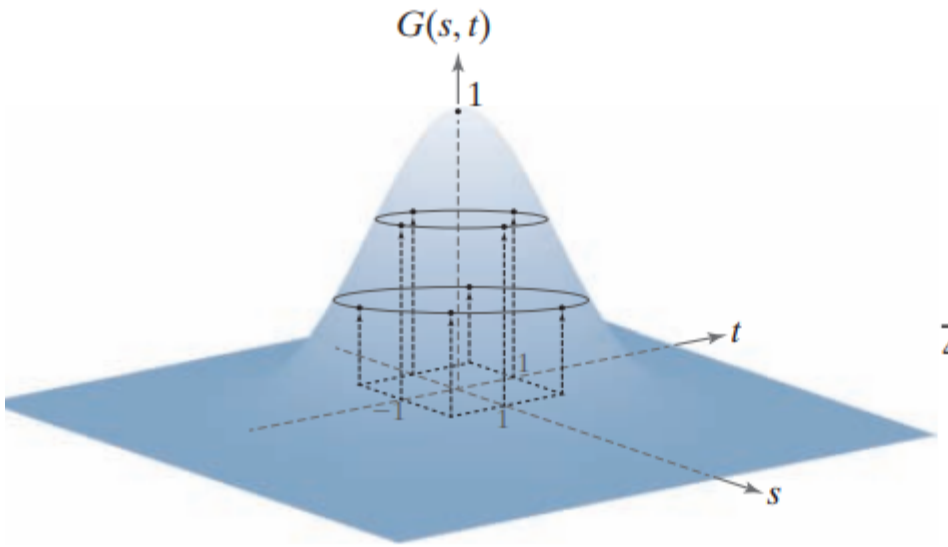
$$r_{\max}^2 = \left[\frac{(m-1)}{2}\sqrt{2}\right]^2 = \frac{(m-1)^2}{2}$$



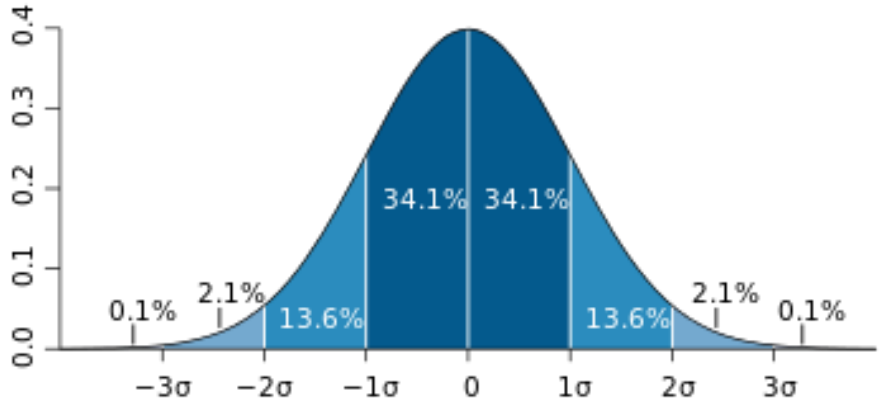
Gaussian filter kernels

(a) Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for K = 1 and σ= 1.

(b) Resulting 3× 3 kernel



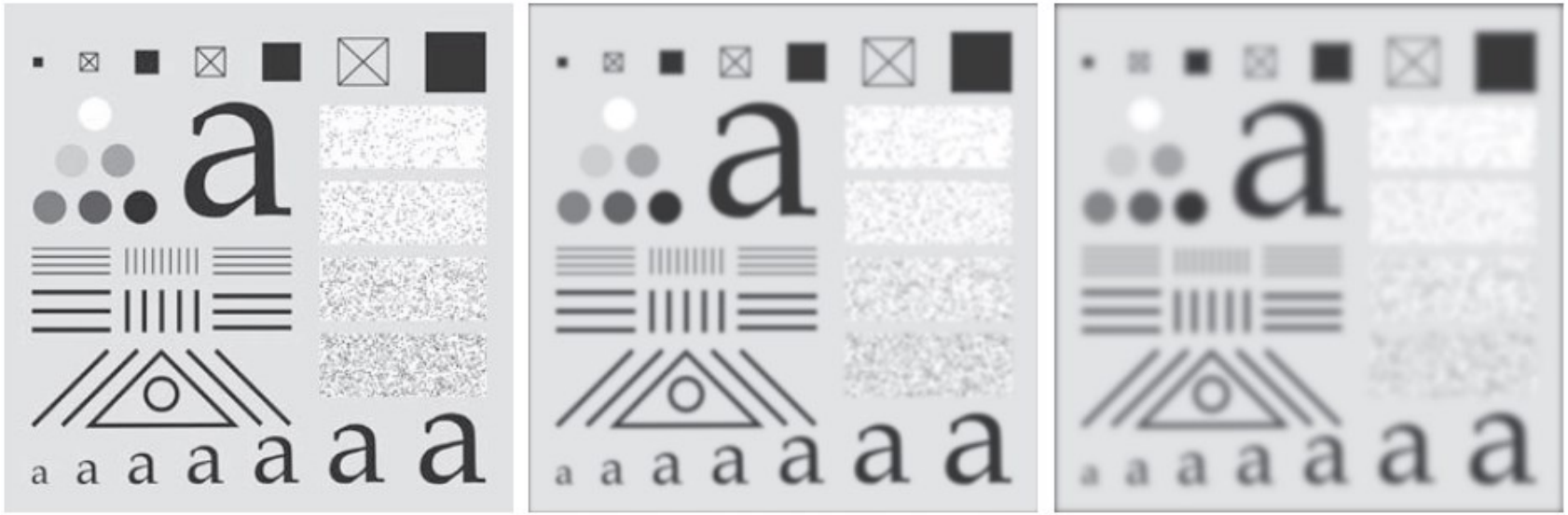
$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$



Gaussian filter kernels

- Lowpass filtering with a Gaussian kernel.

(a) A test pattern of size 1024×1024.
(b) Result of lowpass filtering the pattern with a Gaussian kernel of size 21×21, σ=3.5.
(c) Result of using a kernel of size 43×43, σ=7.
K=1 in all cases.



Lowpass filtering

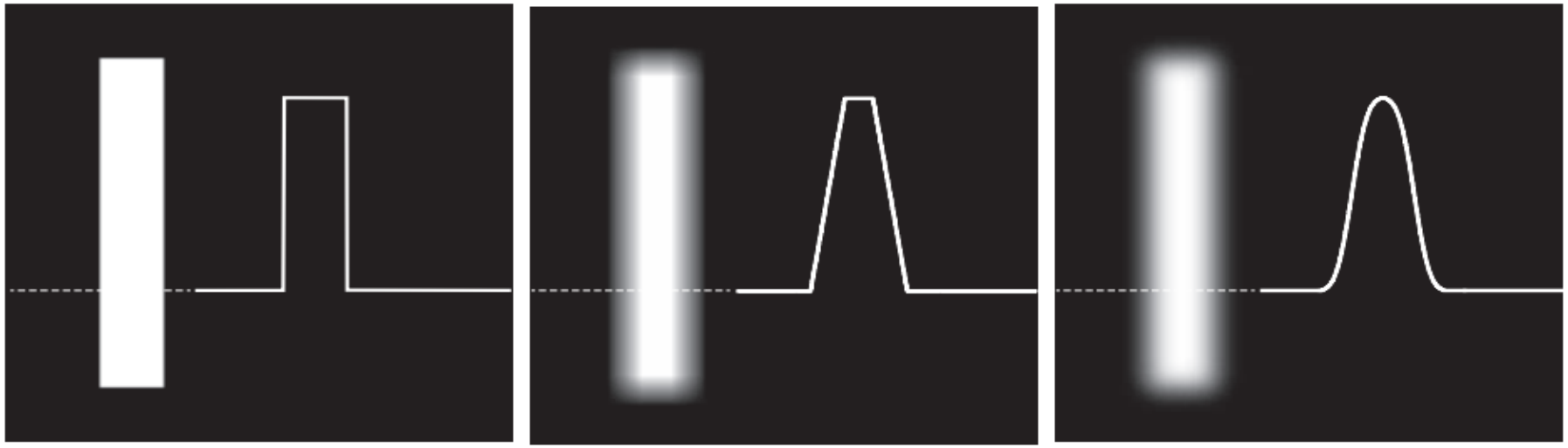
Sự khách nhau giữa Box filter và Gaussian filter

(a) Image of a white rectangle on a black background, and a horizontal intensity profile along the scan line shown dotted.

(b) Result of smoothing this image with a box kernel of size 71×71, and corresponding intensity profile.

(c) Result of smoothing the image using a Gaussian kernel of size 151×151, with K=1 and σ=25.

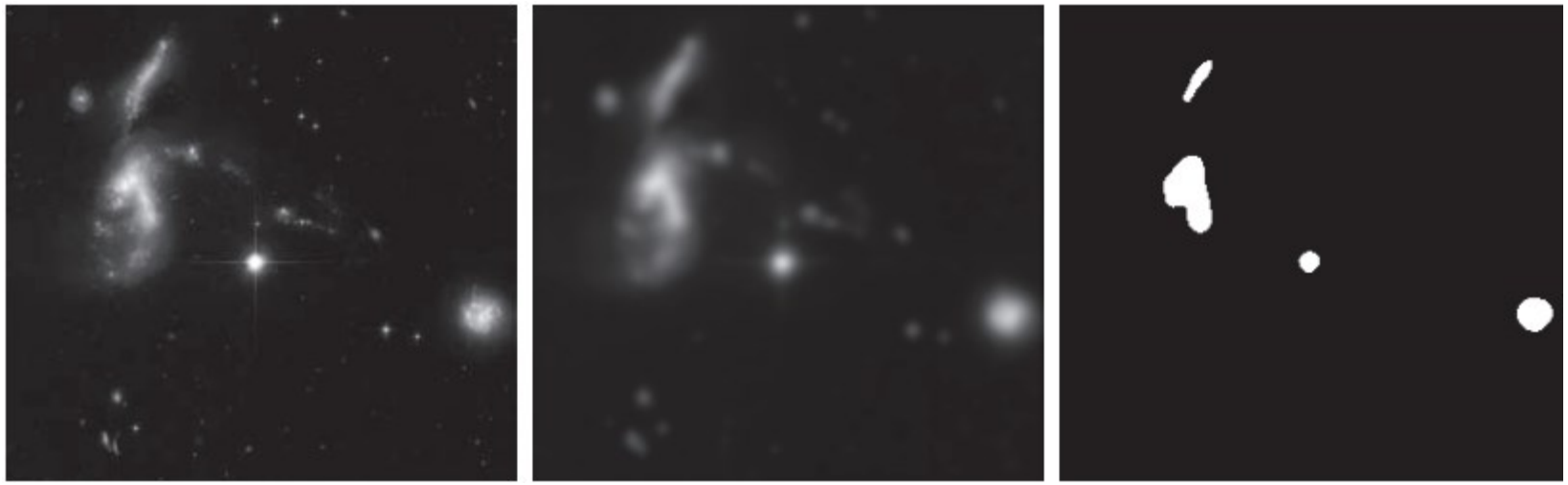
- smoothness of the profile in (c) compared to (b)



Lowpass filtering

- Using lowpass filtering and thresholding for region extraction

(a) A 2566×2758 Hubble Telescope image of the Hickson Compact Group.
(b) Result of lowpass filtering with a Gaussian kernel.
(c) Result of thresholding the filtered image (intensities were scaled to the range [0, 1]). The Hickson Compact Group contains dwarf galaxies that have come together, setting off thousands of new star clusters.



Lowpass filtering

- Shading correction using lowpass filtering.

Shading correction (called flat-field correction) is important because shading is a common cause of erroneous measurements, degraded performance of automated image analysis algorithms, and difficulty of image interpretation by humans

Example: (a) Image shaded by a shading pattern oriented −45° direction
(b) Estimate of the shading patterns obtained using lowpass filtering
(c) Result of dividing (a) by (b).

Order-Statistic (Nonlinear) Filters

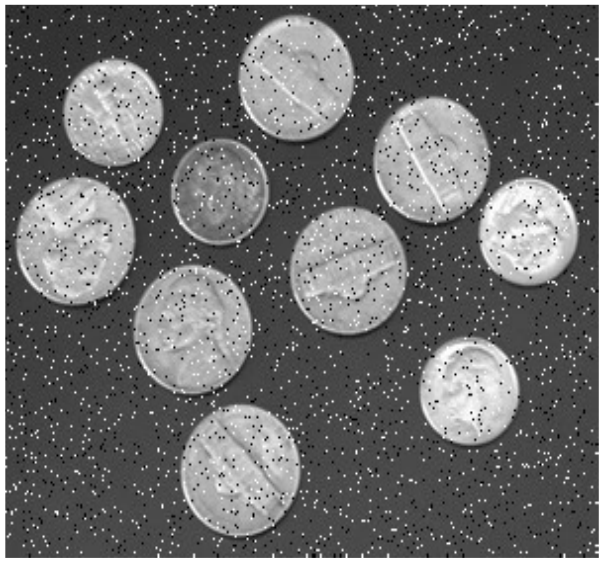
- Response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result.
- Best-known filter is *median filter.*
- Replaces the value of a center pixel by the median of the intensity values in the neighborhood of that pixel.
- Used to remove *impulse or salt-pepper noise*.
- Larger clusters are affected considerably less. ?
- Median represents the 50^{th} percentile of a ranked set of numbers while 100^{th} or 0^{th} percentile results in the so- called *max filter or min filter* respectively.

Nonlinear filters

- The median filter is a non-linear digital filtering technique, often used to remove noise from an image or signal.
- The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the "window", which slides

median filter

Original image

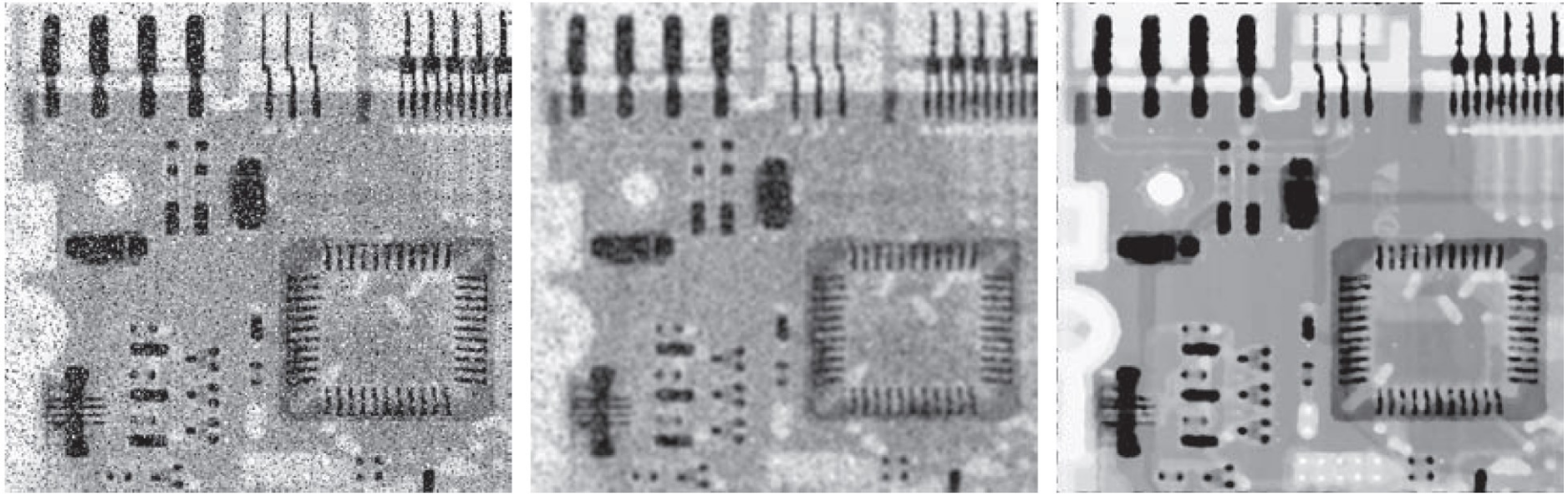


Median filtered image

Nonlinear filters

Example median filter

(a) X-ray image of a circuit board, corrupted by salt-and-pepper noise.
(b) Noise reduction using a 19×19 Gaussian filter with σ=3.
(c) Noise reduction using a 7× 7 median filter.
(Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



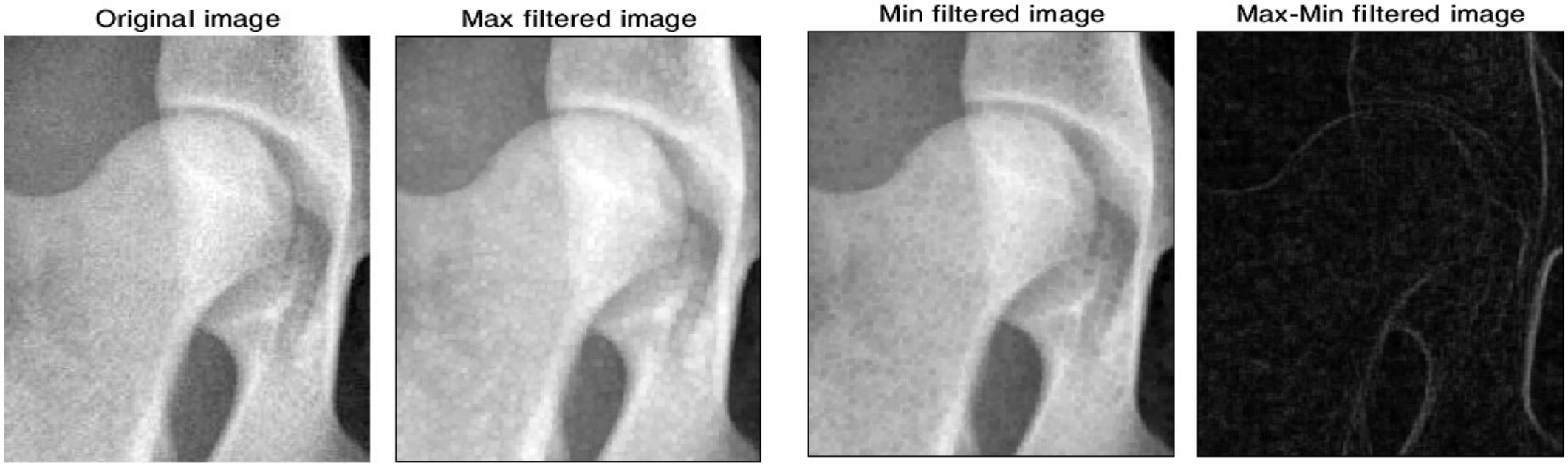
Nonlinear filters

Viết hàm lọc median, max, min, max-min và midpoint

- Max/min filter:

Output using Max/Min values of element within a window size n×n, e.g. 3x3, 7x7….

➢ A new value of pixel is selected from thresholding of surrounding region with referred to Max, Min or Max/Min values.



Nonlinear filters

- Midpoint filter: Return a new value from surrounding region with mean of Min and Max value of filter kernel n×n.

Original image

Midpoint filtered image

Outline

1. Background
2. Basic intensity transformation functions
3. Histogram processing
4. Fundamentals of spatial filtering
5. Smoothing (lowpass) spatial filters
6. Sharpening (highpass) spatial filters

Sharpening Spatial Filters

- Objective of sharpening is to highlight transitions in intensity.
- Uses in printing and medical imaging to industrial inspection and autonomous guidance in military systems.
- Averaging is analogous to integration, so sharpening is analogous to spatial differentiation.
- Thus, image differentiation enhances edges and other discontinuities (such as noise) and deemphasizes areas with slowly varying intensities.

Foundation

- Definition for a first order derivative:

  (1) must be zero in areas of constant intensity

  (2) must be nonzero at the onset of an intensity step or ramp

  (3) must be nonzero along ramps.

- For a second order derivatives:

  (1) must be zero in constant areas

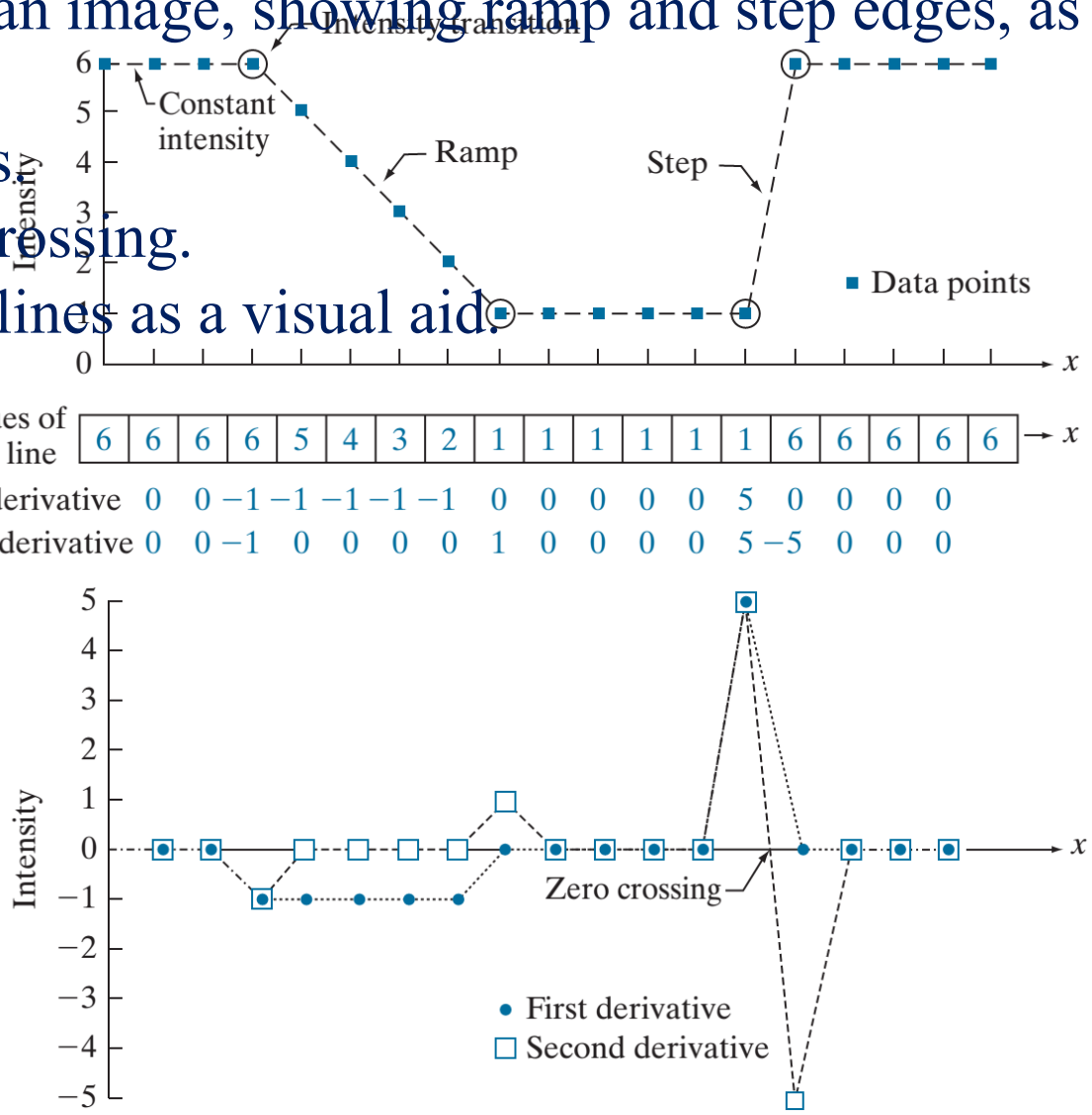
  (2) must be nonzero at the onset

  (3) must be zero along ramps of constant slope.

- First order derivative of a one dimensional function f(x) is the difference of f(x+1) – f(x).
- Second order = f(x+1) + f(x-1) -2f(x)

Foundation

(a) A section of a horizontal scan line from an image, showing ramp and step edges, as well as constant segments.
(b)Values of the scan line and its derivatives.
(c) Plot of the derivatives, showing a zero crossing.
In (a) and (c) points were joined by dashed lines as a visual aid.



Sharpening Spatial Filters

- Using the second derivative for image sharpening—the Laplacian
  - The approach consists of defining a discrete formulation of the second-order derivative and then constructing a filter kernel based on that formulation
  - The simplest isotropic derivative operator is the Laplacian, which, for a function (image) f(x,y) of two variables, is defined:

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

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

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

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Second Derivatives- Laplacian

- Example

(a) Laplacian kernel used to implement equation.
(b) Kernel used to implement an extension of this equation that includes the diagonal terms.
(c) and (d) two other Laplacian kernels.

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Second Derivatives- Laplacian

- The Laplacian for image sharpening is

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

  - where f(x,y) and g(x.y) are the input and sharpened images.
    - ❑ c = −1 if the Laplacian kernels (negative)
    - ❑ c = 1 if either of Laplacian kernels (positive) .

Second Derivatives- Laplacian

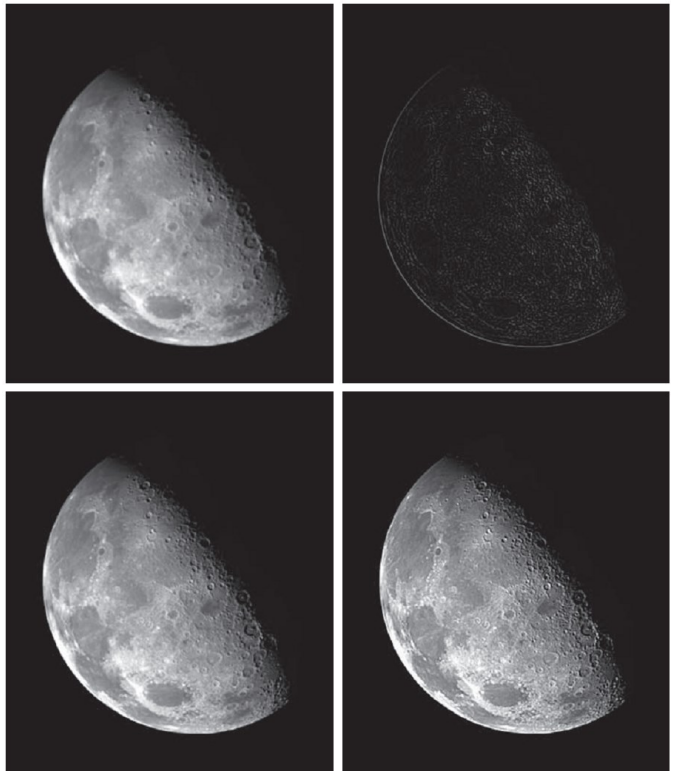
- Example:

a|b
c|d

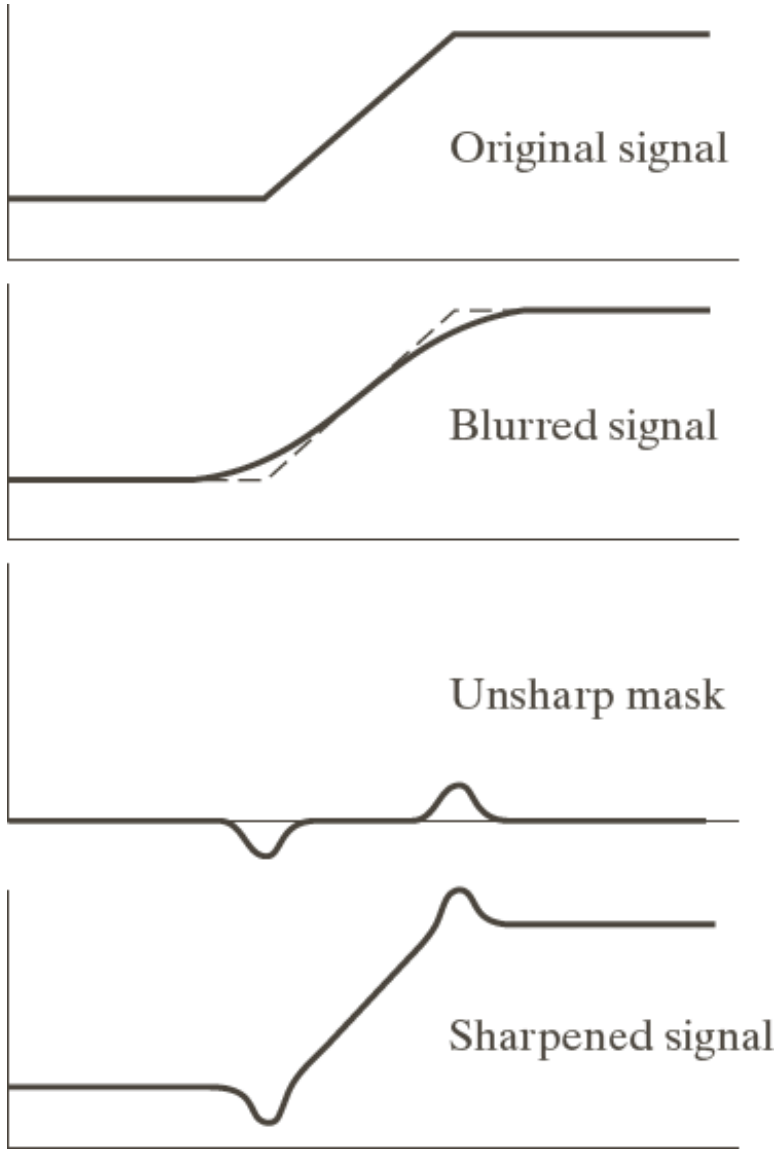
(a) Blurred image of the North Pole of the moon.
(b) Laplacian image obtained using the kernel (a).
(c) Image sharpened (Laplacian kernel negative), g(x,y) with c=−1.
(d) Image sharpened using the same procedure, but with the kernel (b).
(Original image courtesy of NASA)

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1



Unsharp Masking and High boost Filtering



a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

Unsharp Masking and High boost Filtering

- Unsharp Masking
  - Read Original Image f(x,y)
  - Blurred original image f'(x,y)
  - Mask = f(x,y) – f'(x,y)
  - g(x,y) = f(x,y) + Mask
- High Boost Filtering
  - Read Original Image f(x,y)
  - Blurred original image f'(x,y)
  - Mask = f(x,y) – f'(x,y)
  - g(x,y) = f(x,y) + k*Mask, where k>1

Unsharp Masking and High boost Filtering

- **Example**

a
b
c
d
e

FIGURE 3.40
(a) Original image.
(b) Result of blurring with a Gaussian filter.
(c) Unsharp mask. (d) Result of using unsharp masking.
(e) Result of using highboost filtering.

First Derivative –Gradient

- first-order derivatives for image sharpening—the gradient

$$\nabla f \equiv \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$M(x,y) = \|\nabla f\| = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

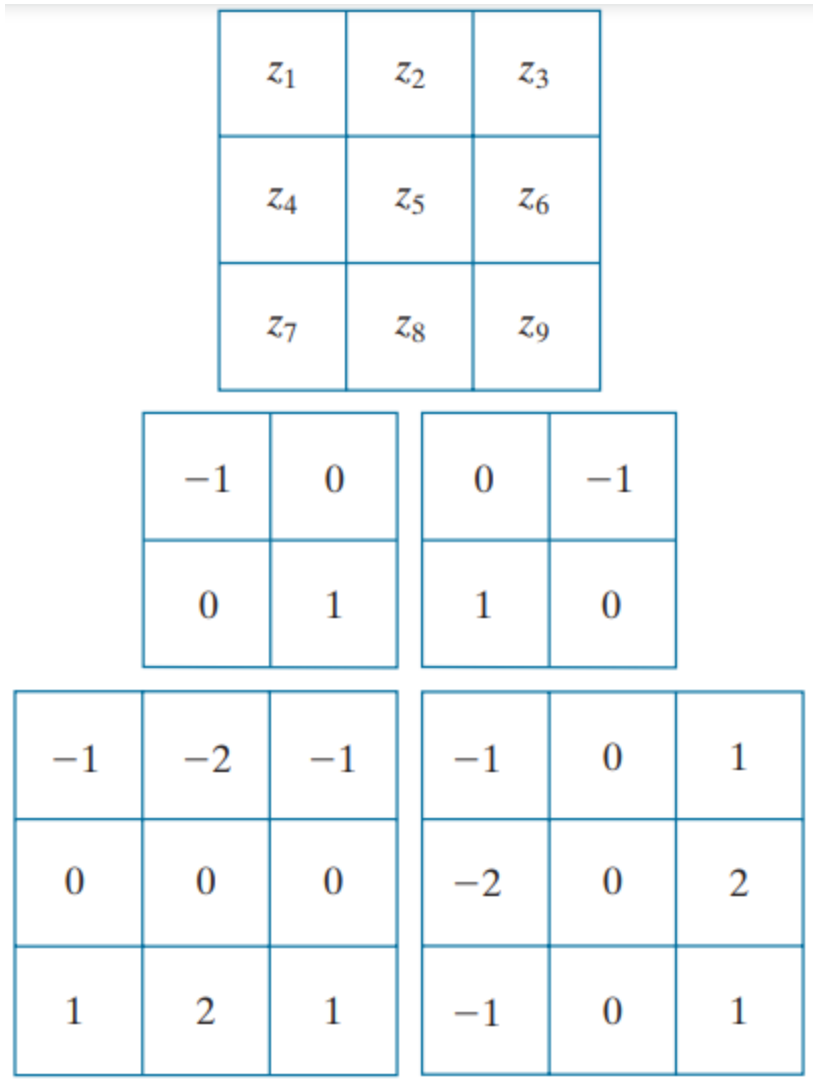
$$M(x,y) \approx |g_x| + |g_y|$$

First Derivative –Gradient

Example Kernel filter:
(a) A 3× 3 region of an image, where the zs are intensity values.
(b)–(c) Roberts cross-gradient operators.
(d)–(e) Sobel operators.
All the kernel coefficients sum to zero, as expected of a derivative operator

prewitt

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$



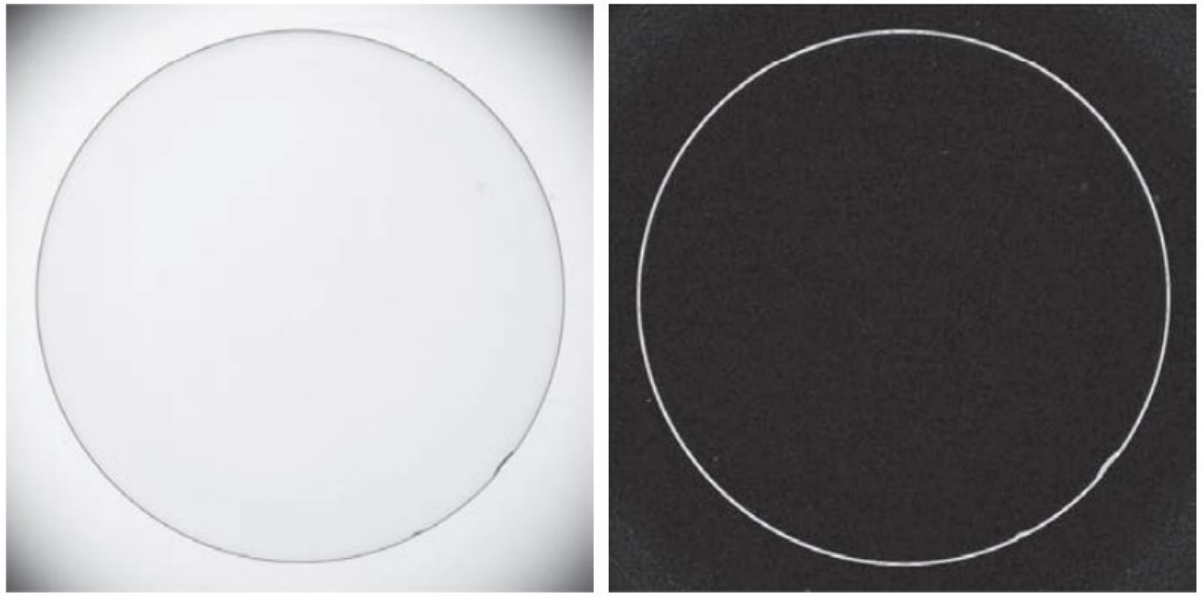
First Derivative –Gradient

Example:

(a) Image of a contact lens (note defects on the boundary at 4 and 5 o'clock).

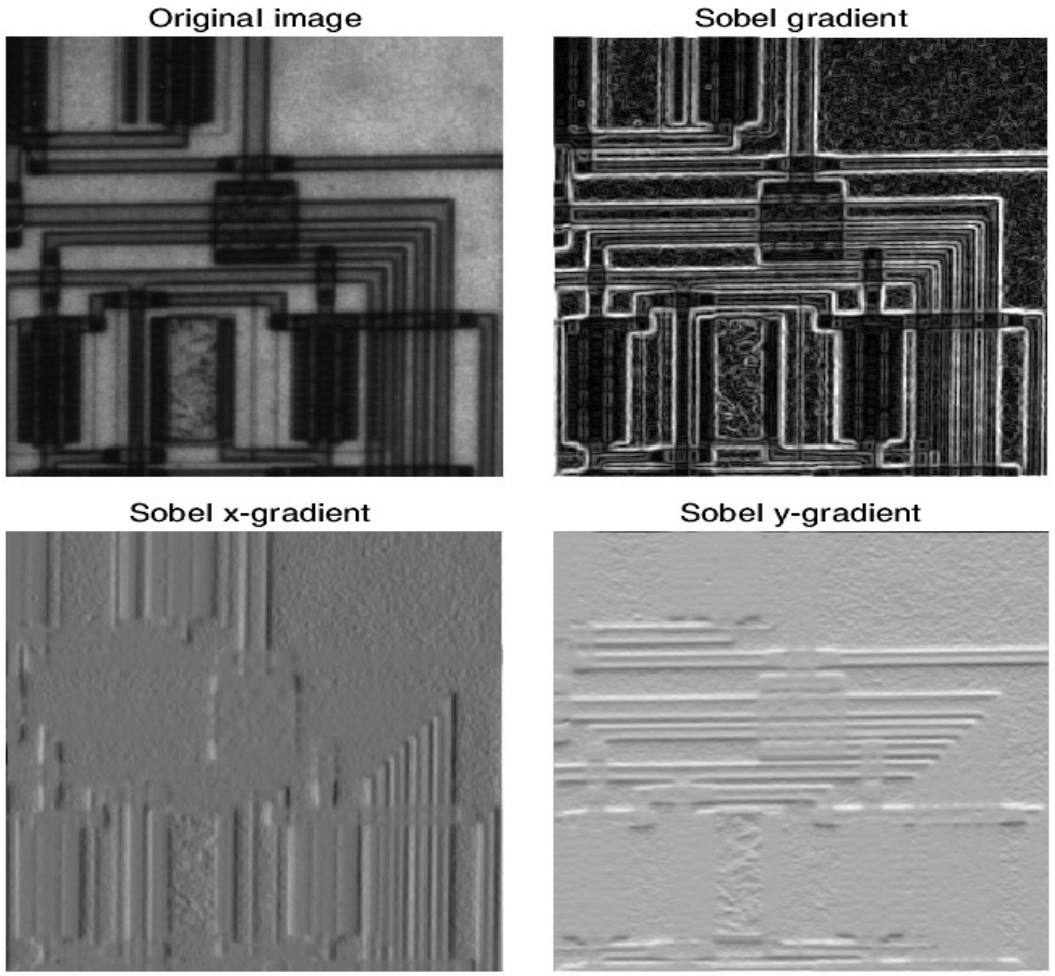
(b) Sobel gradient.

(Original image courtesy of Perceptics Corporation.)



First Derivative –Gradient

- **Example:**



Combining Spatial Enhancement Methods

Applied examples:

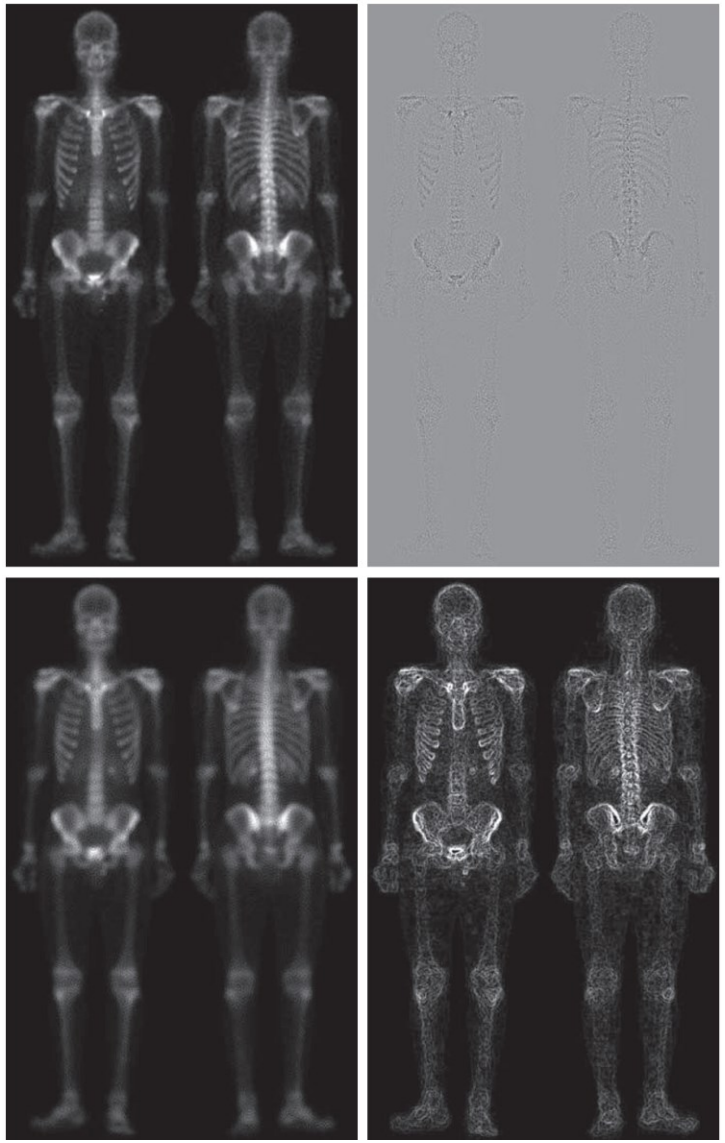
(a) Image of whole body bone scan.

(b) Laplacian of (a).

(c) Sharpened image obtained by adding (a) and (b).

(d) Sobel gradient of image (a).

(Original image courtesy of G.E. Medical Systems.)

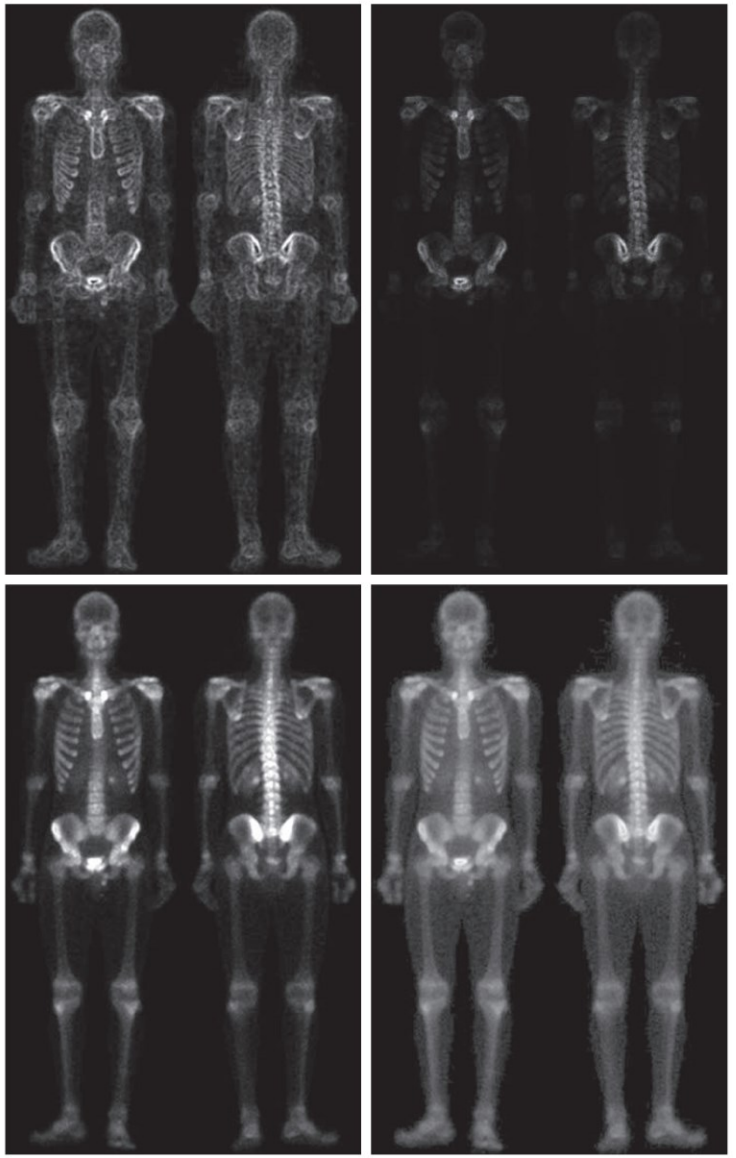


Combining Spatial Enhancement methods

Example (continued)

(e) Sobel image smoothed with a 5×5 box filter.
(f) Mask image formed by the product of (b) and (e).
(g) Sharpened image obtained by the adding images (a) and (f).
(h) Final result obtained by applying a powerlaw transformation to (g).

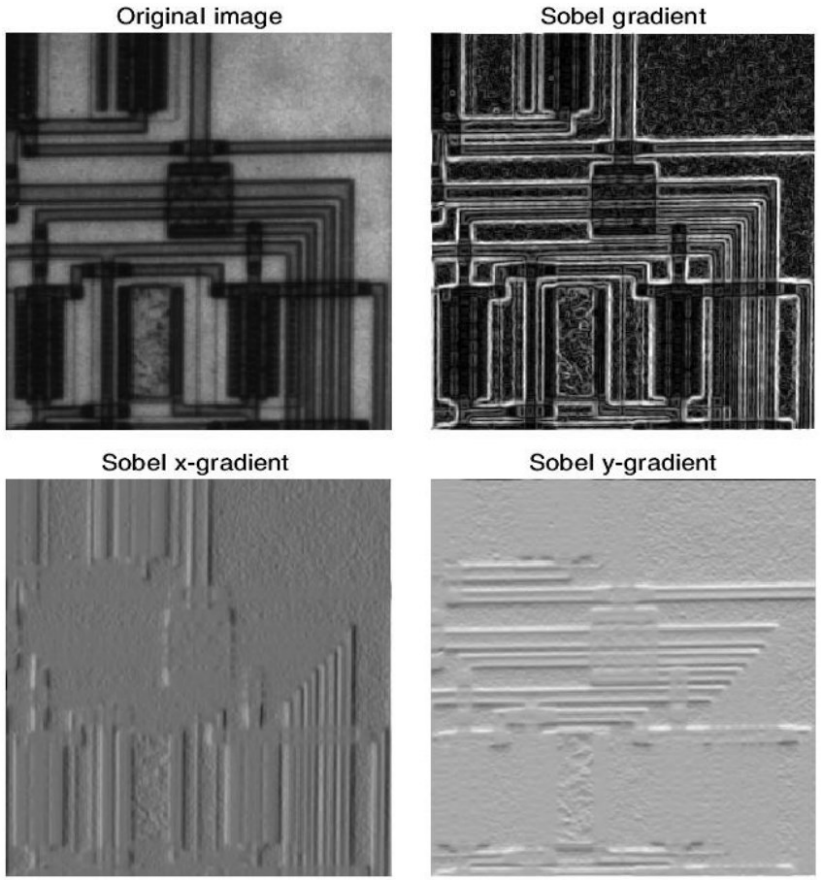
➢ Compare images (g) and (h) with (a).



Some filter methods

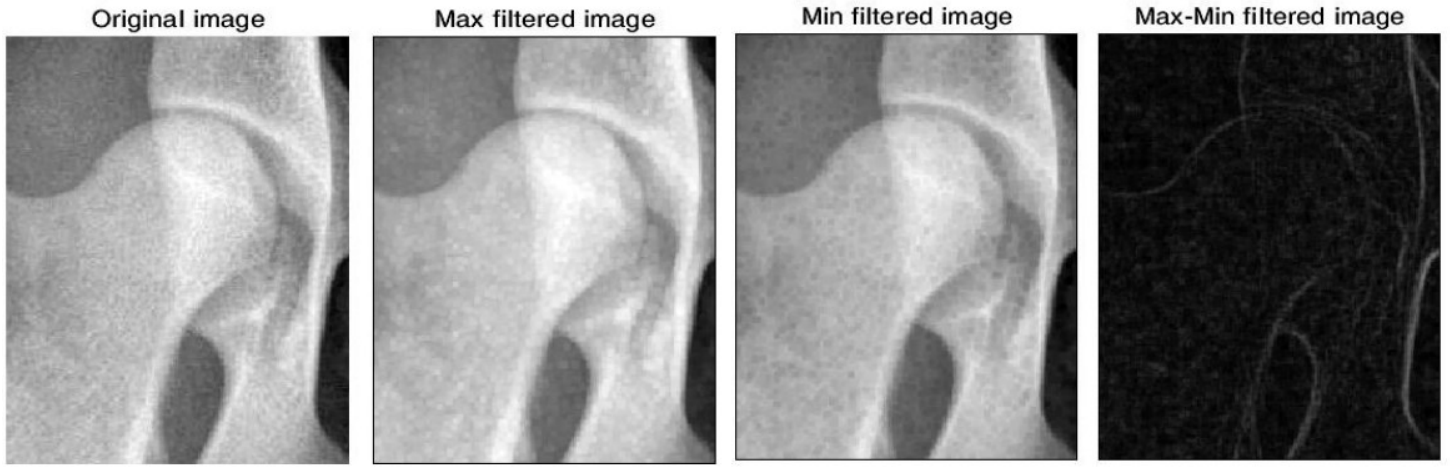
$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ và } k_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

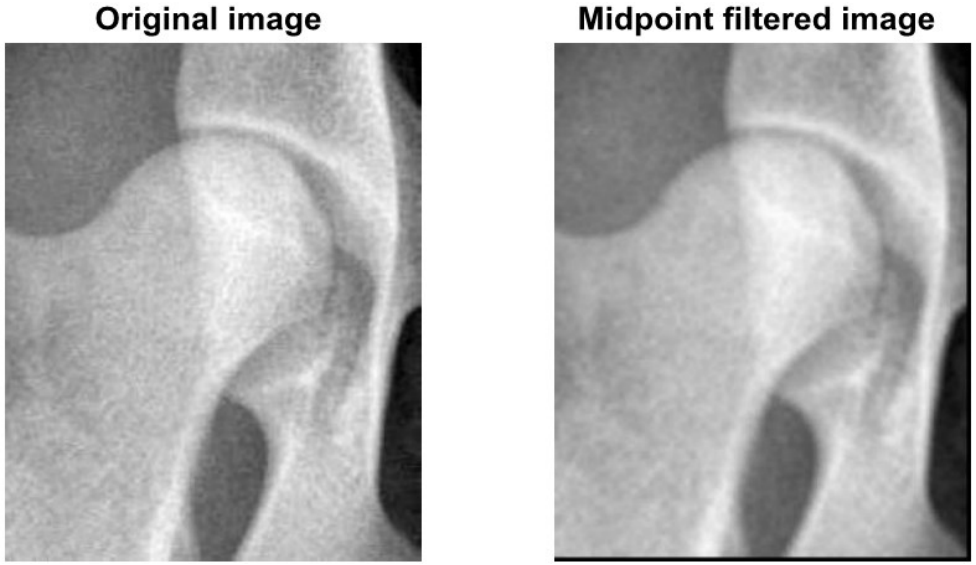
```
B = imread('circuit.tif' );
kx = [-1 0 1; -2 0 2; -1 0 1];
ky=kx';
Gx=conv2(B,kx, 'same' );
Gy=conv2(B,ky, 'same' );
G=sqrt(Gx.^2+Gy.^2);
imshow('Original image',B);
imshow('Sobel gradient',G);
imshow('Sobel x-gradient',Gx);
imshow('Sobel y-gradient',Gy);
```



Some filter methods

- Median filter
- Gaussian filter
- ***max/ min***
- ***midpoint***





Exercises

- See the supplementary file

Thanks for your attention!

Q&A

Appendix: Bài tập

1. Hãy đọc vào một ảnh từ bộ nhớ ngoài, thực hiện lần lượt các phép lọc trung bình, lọc Sobel trên ảnh gốc với kích thước bộ lọc 3×3 sau đó hiển thị ảnh gốc và các ảnh sau khi lọc?

2. Hãy đọc vào một ảnh từ bộ nhớ ngoài, thực hiện các phép lọc trung vị, Gaussian, max/min trên ảnh gốc với kích thước bộ lọc 7×7 sau đó hiển thị ảnh gốc và các ảnh thu được của mỗi phương pháp lọc?

3. Vì sao hàm lọc trung bình là một hàm tuyến tính nhưng hàm lọc trung vị lại là hàm phi tuyến?

4. Thực hiện thay đổi độ tương phản ảnh bằng cách dùng phương pháp biến đổi độ tương phản: Cho các tham số nằm trong khoảng $[a, b]$ và $[z_1, z_k]$, một hình ảnh có độ tương phản yếu, thực hiện các biến đổi độ tương phản và làm các thực nghiệm để thấy tính hiệu quả của việc sử dụng độ tương phản như làm rõ đối tượng, làm mờ đối tượng,.. và cho nhận xét kết quả?

Bài tập

5. So sánh tính chất của hàm trung bình và hàm trung vị. Trường hợp nào thì nên sử dụng sử dụng lọc trung bình, lọc trung vị?

6. Phương pháp lọc Gaussian thường được xem là phương pháp trung bình có trọng số, tại sao?

7. Thực hiện lọc làm mịn ảnh bằng phương pháp Gaussian. Thực hiện lọc làm mịn cho một hình ảnh bằng cách lựa chọn vài giá trị độ lệch chuẩn σ và vài kích thước lọc khác nhau. Xem xét mức độ làm mịn tương ứng với từng kích thước lọc và giá trị độ lệch chuẩn khác nhau, làm sao để chọn đúng giá trị σ và kích thước bộ lọc phù hợp?

8. Viết chương trình đọc vào một ảnh, sau đó áp dụng từng phép lọc Gaussian, Sobel, Trung bình, Min, Max, Midpoint riêng biệt trên ảnh gốc, sau đó hiển thị ảnh kết quả của mỗi phép lọc cùng ảnh gốc và cho nhận xét?

Bài tập

9. Viết chương trình đọc vào một ảnh, sau đó thêm nhiễu muối hạt tiêu vào ảnh được gọi là ảnh gốc nhiễu, áp dụng từng phép lọc Gaussian, Trung bình, Trung vị riêng biệt trên ảnh gốc nhiễu, hiển thị kết quả của mỗi phép lọc cùng với ảnh gốc nhiễu và cho nhận xét?

10. Viết chương trình đọc vào một ảnh, thực hiện lọc Gaussian lần lượt với độ lệch chuẩn là σ=1, σ=1.43, σ=2, σ=2.73 riêng biệt trên ảnh gốc nhiễu, sau đó hiển thị ảnh kết quả của mỗi lần lọc cùng với ảnh gốc và cho nhận xét?

11. Viết chương trình đơn giản để thay đổi các mức cân bằng màu của ảnh bằng cách áp dụng mỗi giá trị màu bởi những hệ số cân bằng màu khác nhau (có thể lập trình để thay đổi cân bằng màu thông qua thanh trượt). Cho biết kết quả thay đổi như thế nào sau mỗi khi áp dụng các mức cân bằng màu khác nhau?

12. Viết chương trình tính lược đồ mức xám của một ảnh và tiến hành cân bằng sáng bằng lược đồ mức xám theo các giá trị cân bằng khác nhau?