Industrial Computer Vision

- Frequency-based Image Filtering

4^{th} lecture, 2022.09.28

Lecturer: Youngbae Hwang

Contents

- Sobel filter for image gradient
- Unsharp mask for image sharpening
- Discrete Fourier Transform
- Frequency-domain Image Filtering
- Image Thresholding
- Morphological Filter

1st Derivative filtering

- Implementing 1st derivative filters is difficult in practice
- For a function f(x, y) the gradient of f at coordinates (x, y) is given as the column vector:

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

1st Derivative filtering

- The magnitude of this vector is given by:

$$\nabla f = mag(\nabla \mathrm{f}) = \left[G_x^2 + G_y^2\right]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

- For practical reasons this can be simplified as:

$$\nabla f \approx \left|G_x\right| + \left|G_y\right|$$

Gradient Mask

- Roberts cross-gradient operators, 2x2

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$G_x = (z_9 - z_5) \quad \text{and} \quad G_y = (z_8 - z_6)$$

$$\nabla f = [G_x^2 + G_y^2]^{1/2} = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2}$$

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

−1	0
0	1

0	−1
1	0

Gradient Mask

- Sobel operators, 3 x 3
- There is some debate as to how best to calculate these gradients but we will use:

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

- which is based on these coordinates

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Sobel operator

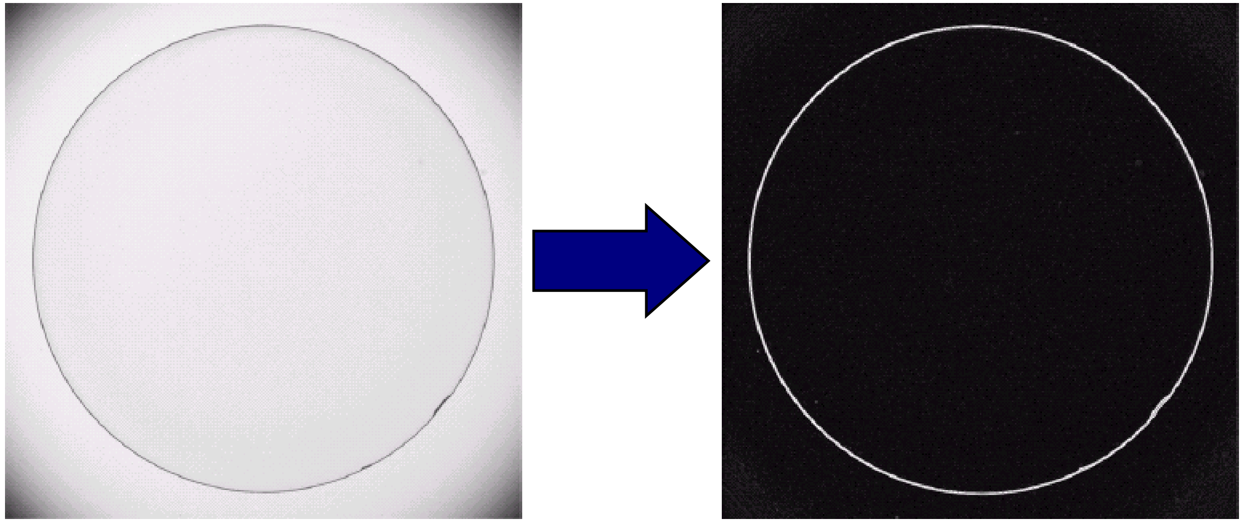
- Based on the previous equations we can derive the Sobel Operators

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

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

- To filter an image it is filtered using both operators the results of which are added together

Sobel example



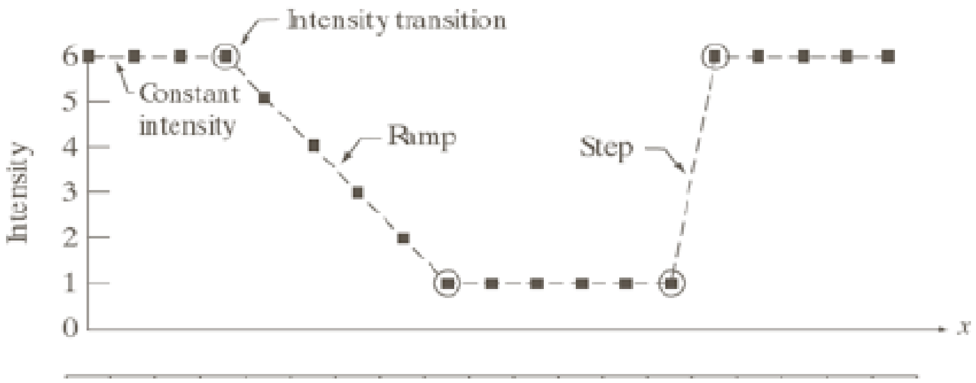
An image of a contact lens which is enhanced in order to make defects (at four and five o’clock in the image) more obvious

- Sobel filters are typically used for edge detection

Sharpening filters

Sharpening filters:

Enhance transitions in intensity.



Constant regions, ramps and steps.

Ramp: joins 2 regions of constant intensity by several pixels
Step: joins 2 regions of constant intensity by 2 pixels.
Onset: the set of transition pixels

Unsharp masking and highboost filtering:

1. Blur original image

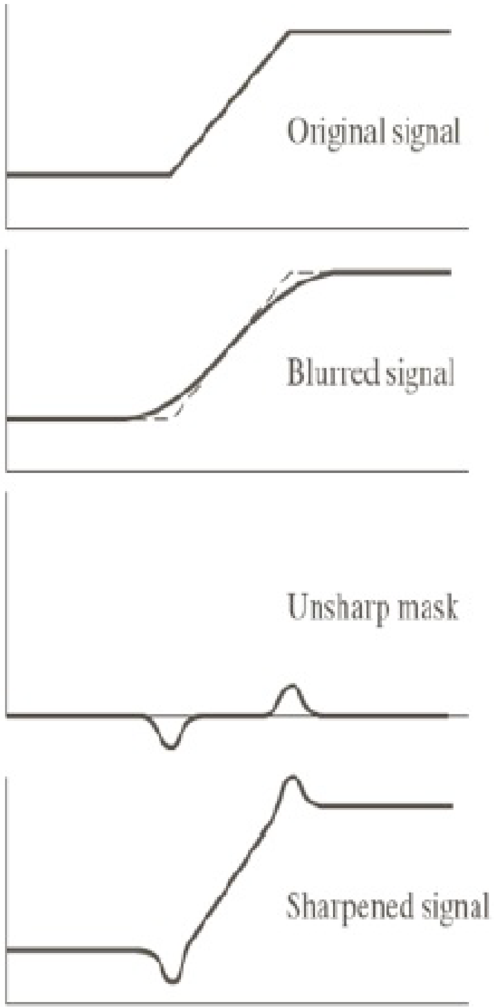
$$f(x,y) \rightarrow \bar{f}(x,y)$$

2. Subtract blurred from image to create a mask

$$g_{\text{mask}}(x,y) = f(x,y) - \bar{f}(x,y)$$

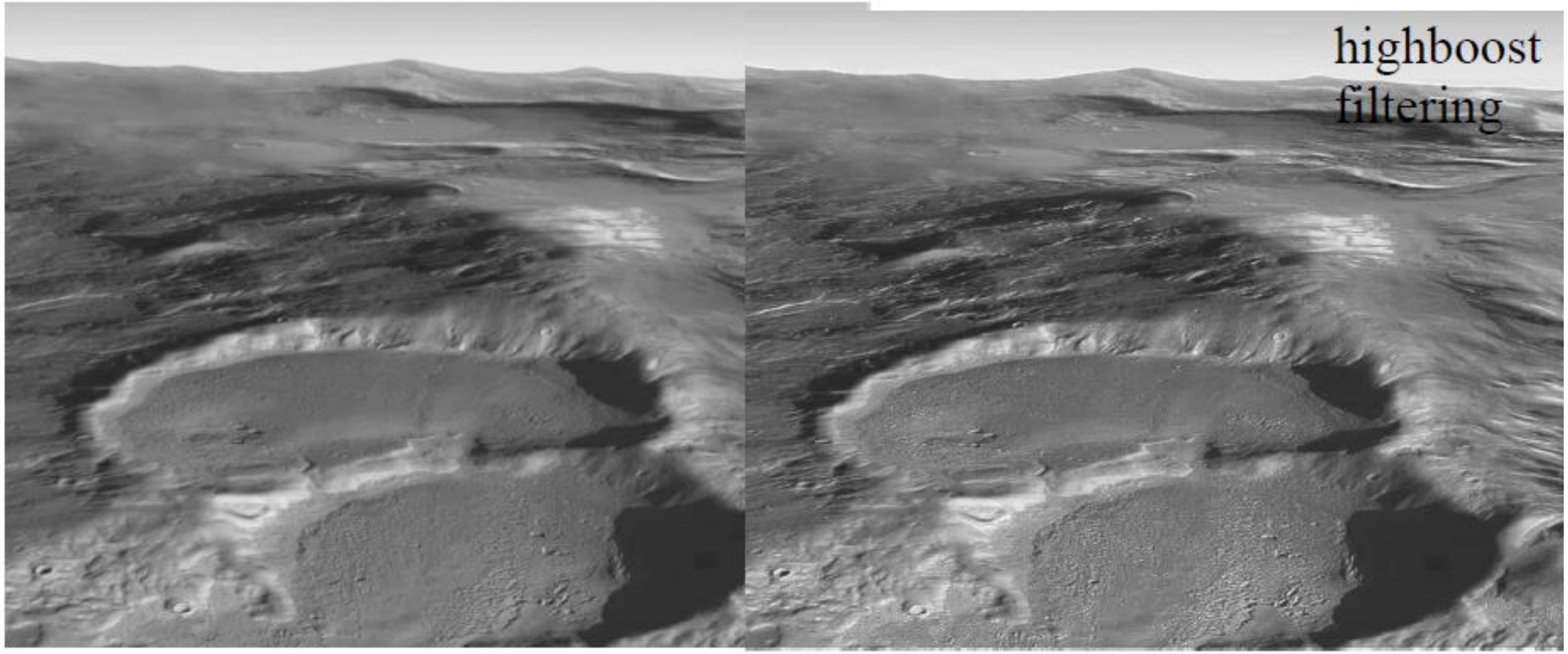
3. Add the mask to the original

$$g(x,y) = f + k * g_{\text{mask}}(x,y)$$



Highboost filtering

- The global effect will be that of enhancing the edges.

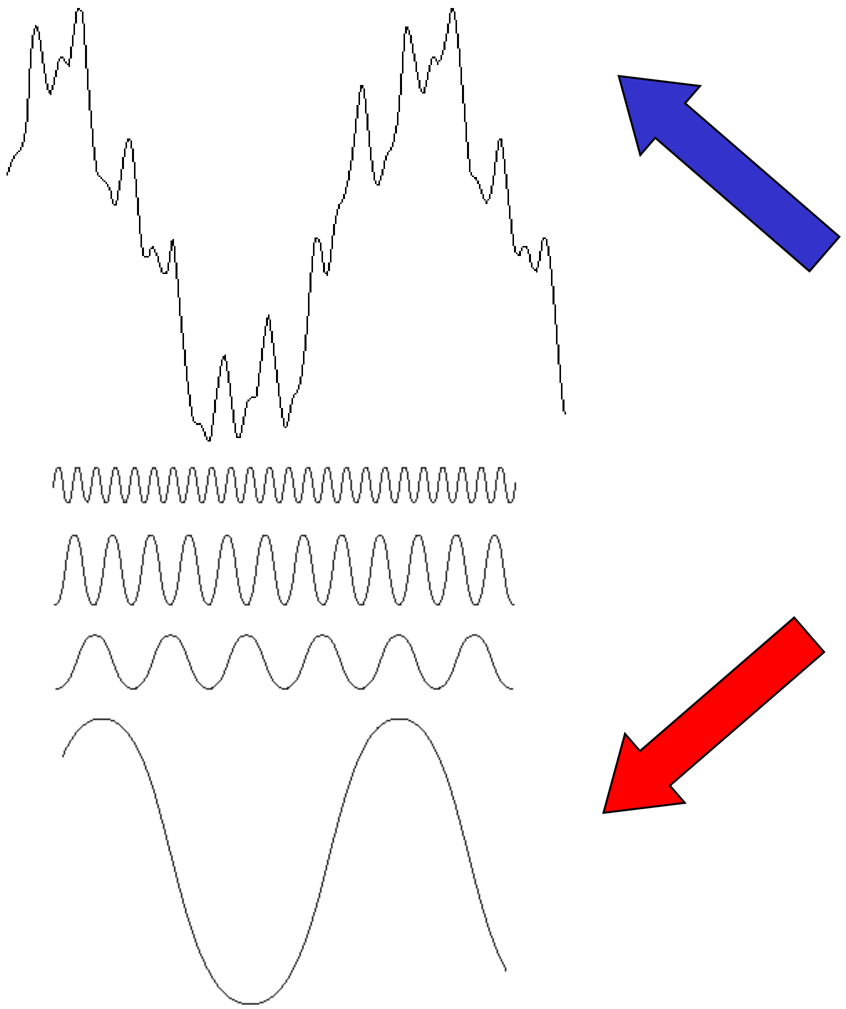


The result of unsharp masking

- (a) Unretouched “soft-tone” digital image (b) Image blurred using a 31 x 31 Gaussian lowpass filter with σ = 5. (c) Mask. (d) Result of unsharp masking using Eq. (3-65) with k = 1. (e) and (f) Results of highboost filtering with k = 2 and k = 3, respectively.

a b c
d e f

Background: Fourier Series

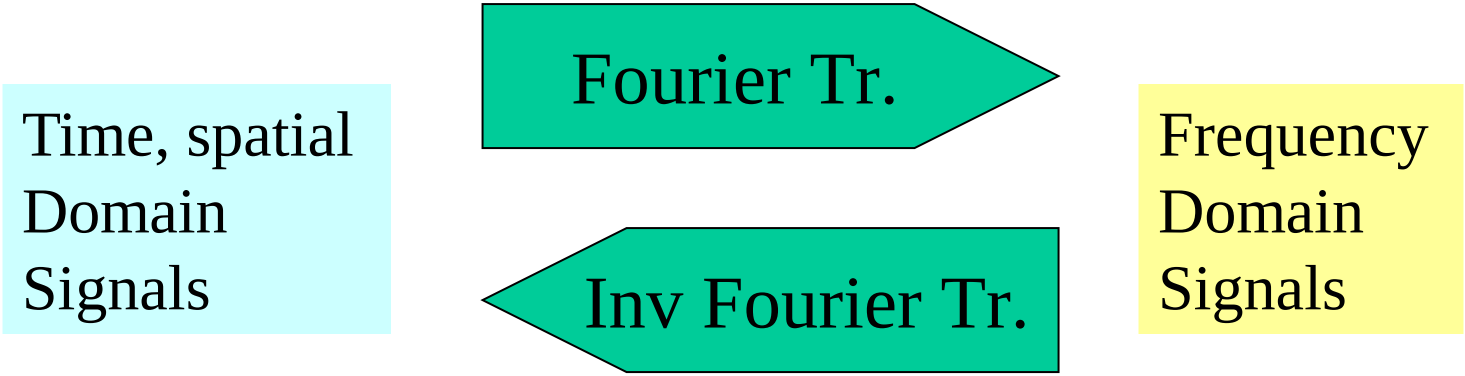


Fourier series:

Any periodic signals can be viewed as weighted sum of sinusoidal signals with different frequencies

Frequency Domain: view frequency as an independent variable

Fourier Tr. and Frequency Domain



1-D, Continuous case

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

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

Fourier Tr. and Frequency Domain (cont.)

1-D, Discrete case

Fourier Tr.: $$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{-j2\pi ux/M}$$ u = 0,…,M-1

Inv. Fourier Tr.: $$f(x) = \sum_{u=0}^{M-1} F(u)e^{j2\pi ux/M}$$ x = 0,…,M-1

$F(u)$ can be written as

$$F(u) = R(u) + jI(u) \quad \text{or} \quad F(u) = |F(u)|e^{-j\phi(u)}$$

where

$$|F(u)| = \sqrt{R(u)^2 + I(u)^2} \qquad \phi(u) = \tan^{-1}\left(\frac{I(u)}{R(u)}\right)$$

2-Dimensional Discrete Fourier Transform

- For an image of size MxN pixels

2-D DFT

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

u = frequency in x direction, $u = 0, \ldots, M-1$
v = frequency in y direction, $v = 0, \ldots, N-1$

2-D IDFT

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

$x = 0, \ldots, M-1$
$y = 0, \ldots, N-1$

2-Dimensional Discrete Fourier Transform (cont.)

- $F(u,v)$ can be written as

$$F(u,v) = R(u,v) + jI(u,v) \quad \text{or} \quad F(u,v) = |F(u,v)|e^{-j\phi(u,v)}$$

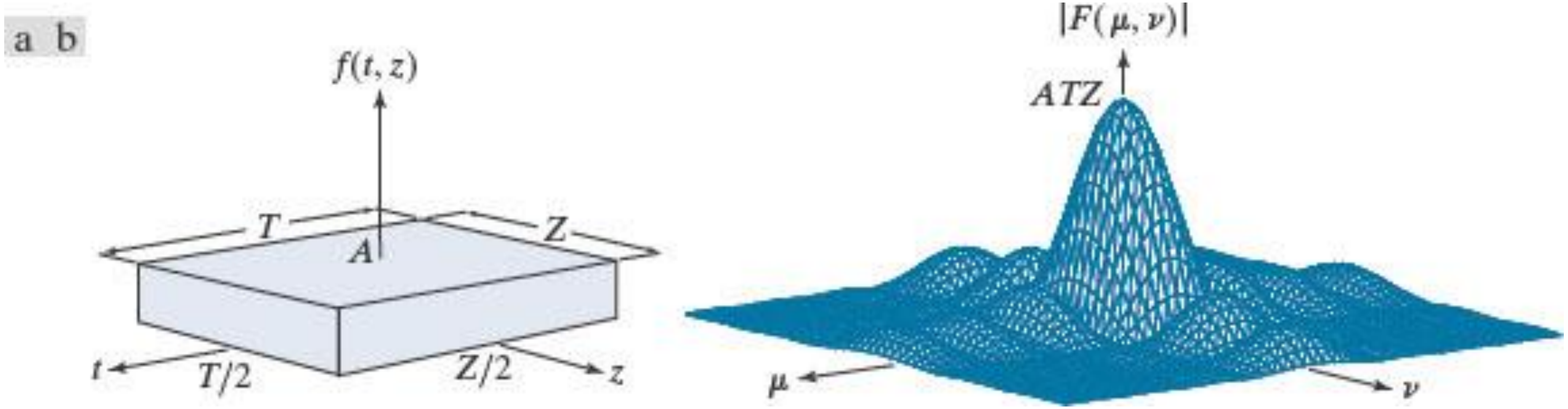
where

$$|F(u,v)| = \sqrt{R(u,v)^2 + I(u,v)^2} \qquad \phi(u,v) = \tan^{-1}\left(\frac{I(u,v)}{R(u,v)}\right)$$

For the purpose of viewing, we usually display only the Magnitude part of $F(u,v)$

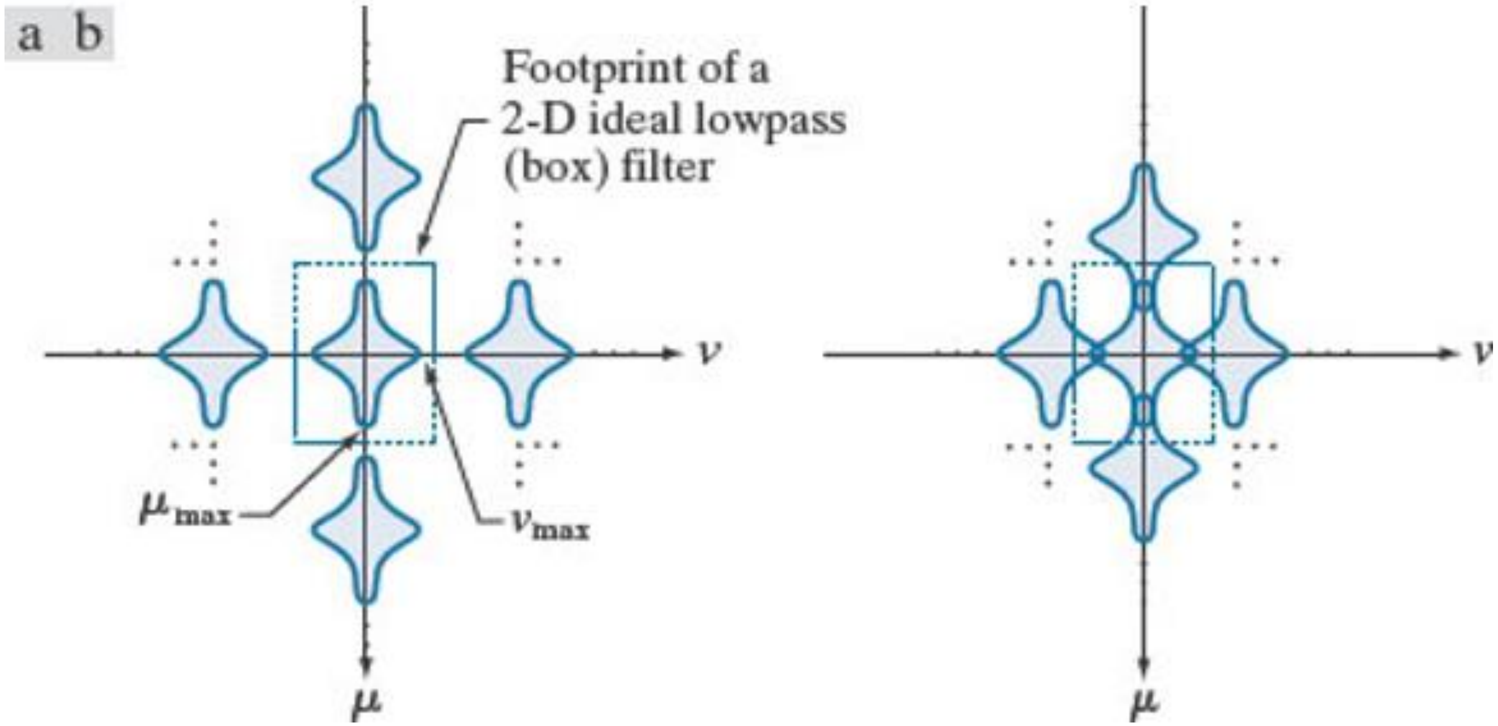
2D DFT of box filter

- (a) A 2-D function and (b) a section of its spectrum. The box is longer along the **t**-axis, so the spectrum is more contracted

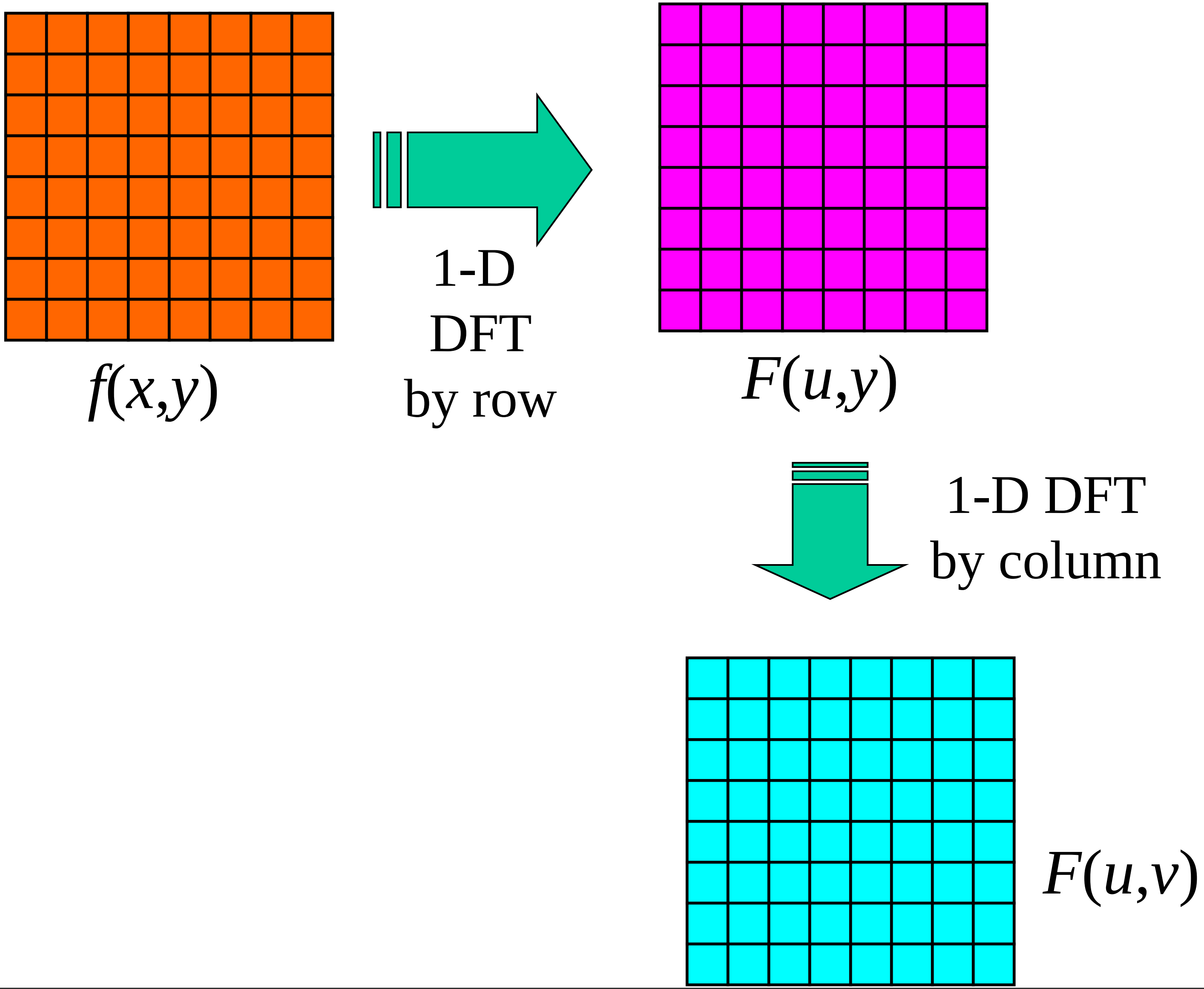


2D DFT of over- and under- sampled function

- Two-dimensional Fourier transforms of (a) an over-sampled, and (b) an under-sampled, band-limited function.

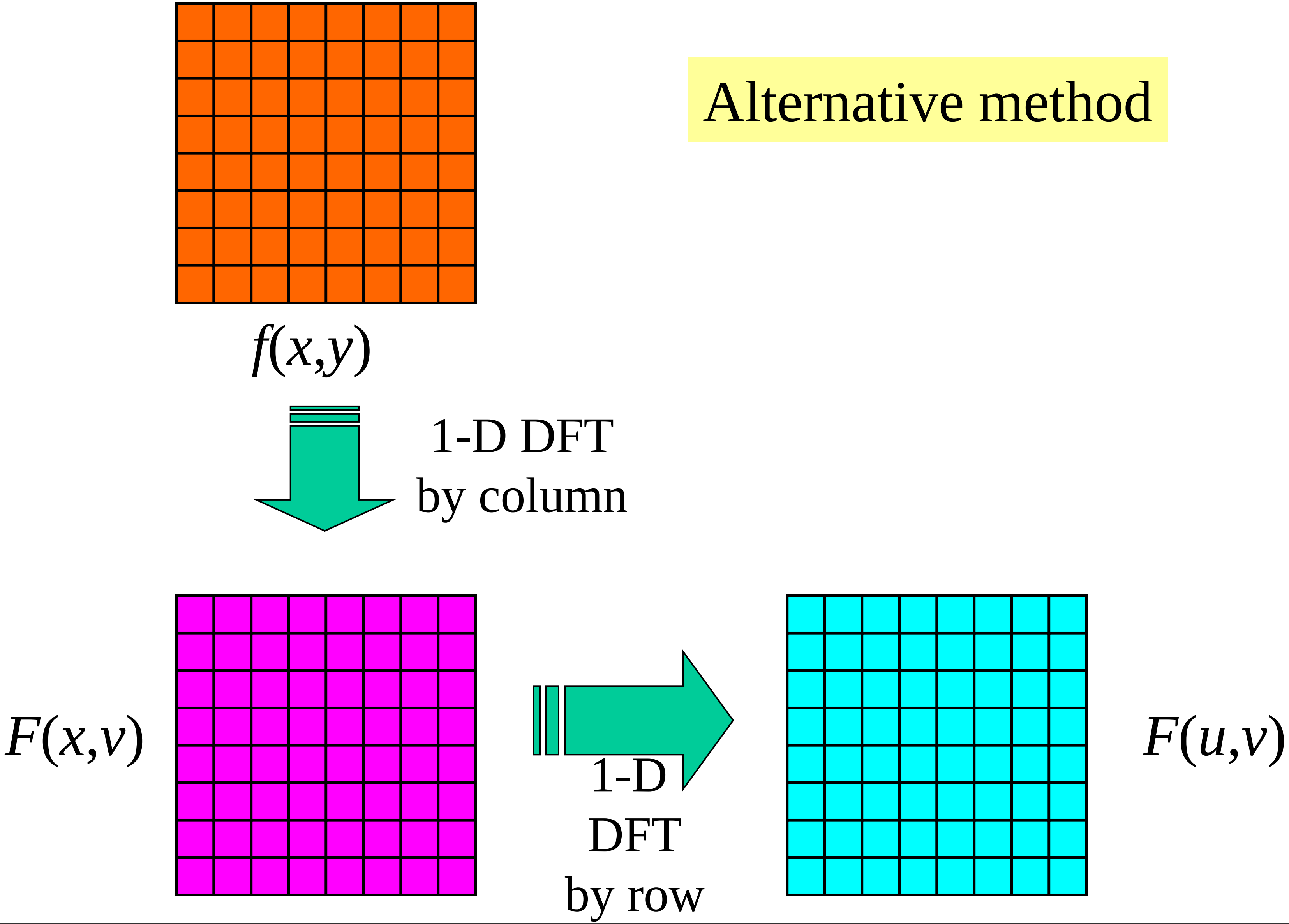


How to Perform 2-D DFT by Using 1-D DFT



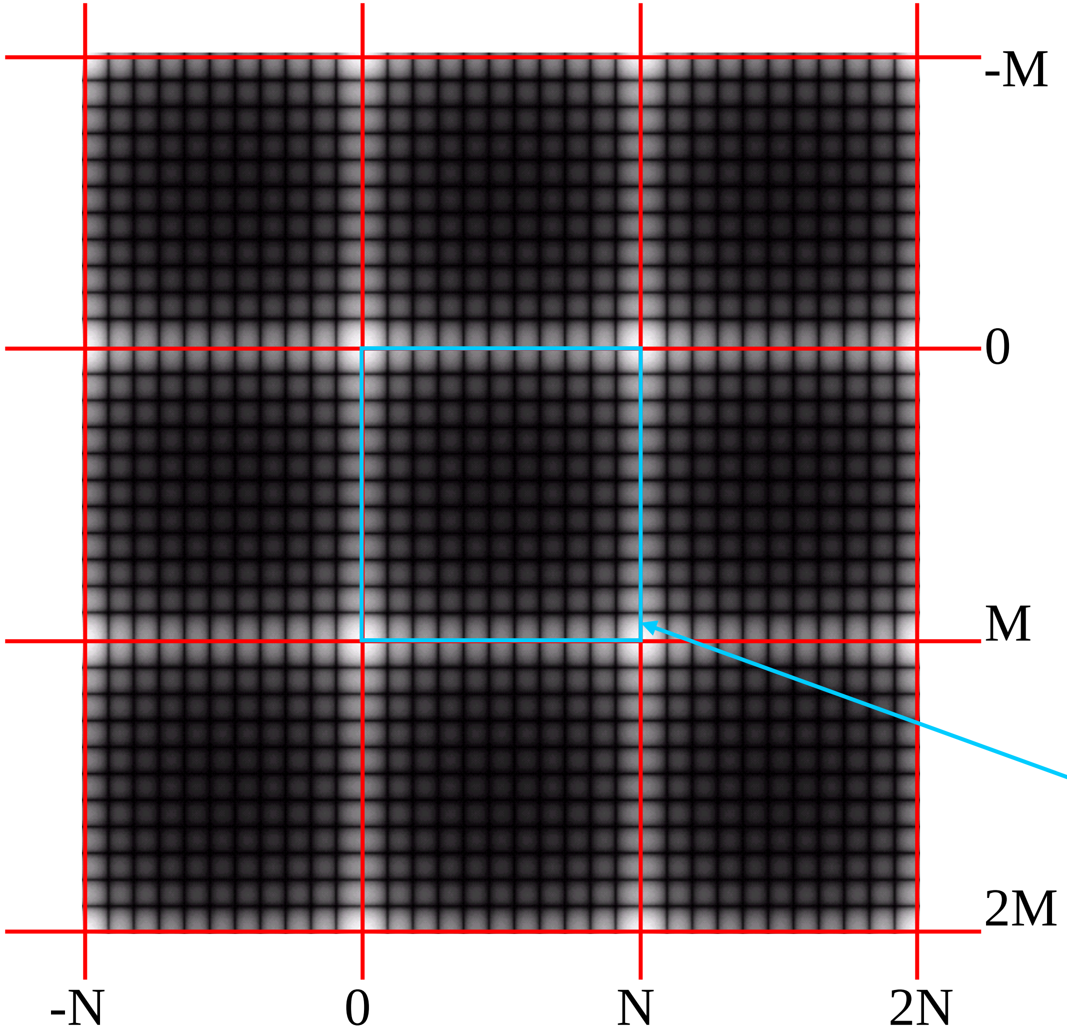
How to Perform 2-D DFT by Using 1-D DFT (cont.)

Alternative method



Periodicity of 2-D DFT

2-D DFT: $F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$



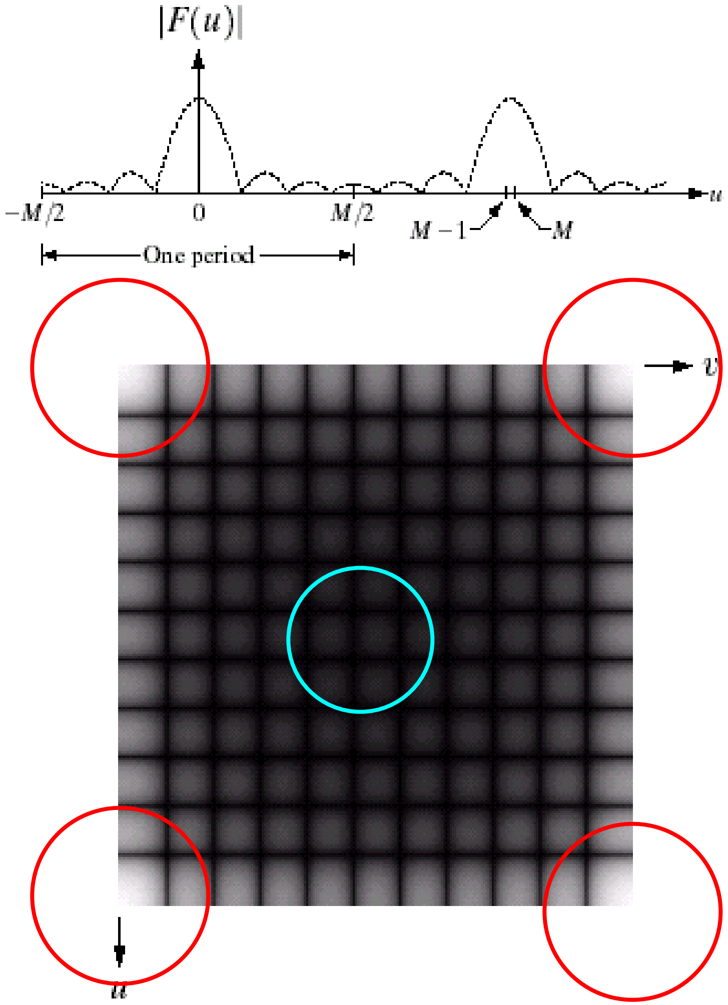
$g(x,y)$

For an image of size NxM pixels, its 2-D DFT repeats itself every N points in x-direction and every M points in y-direction.

We display only in this range

Conventional Display for 2-D DFT

$F(u,v)$ has low frequency areas at corners of the image while high frequency areas are at the center of the image which is inconvenient to interpret.



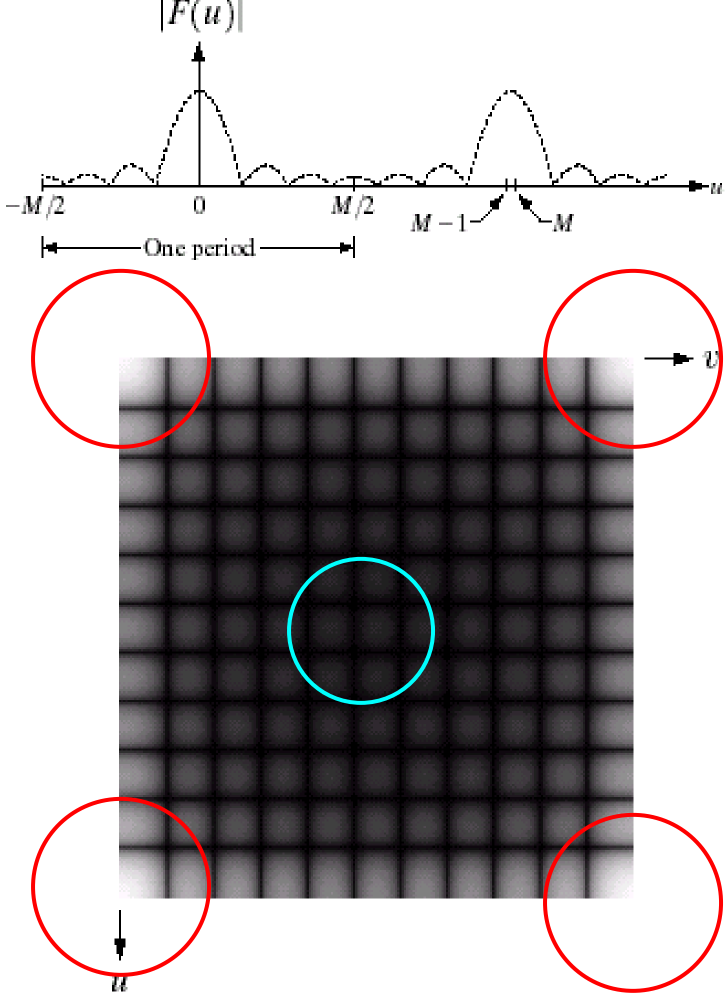
High frequency area

Low frequency area

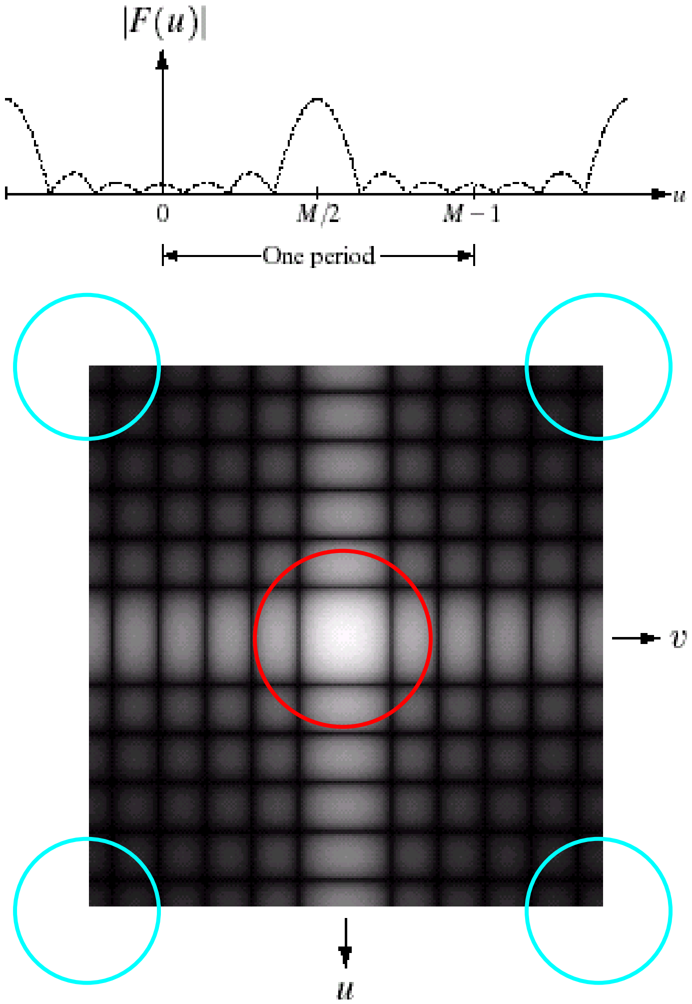
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

2-D FFT Shift : Better Display of 2-D DFT

2-D FFT Shift is a MATLAB function: Shift the zero frequency of $F(u,v)$ to the center of an image.



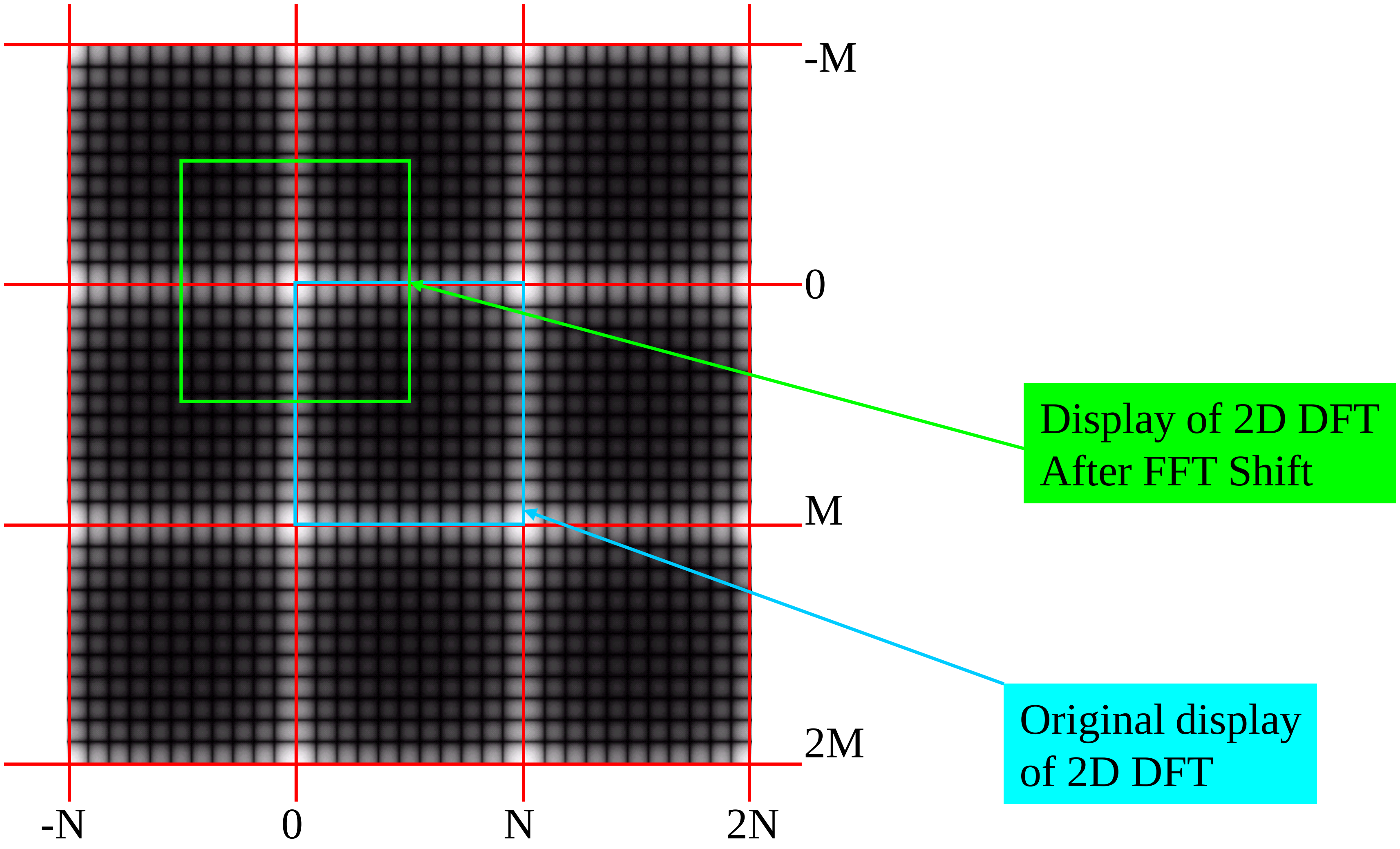
2D FFTSHIFT



High frequency area

Low frequency area

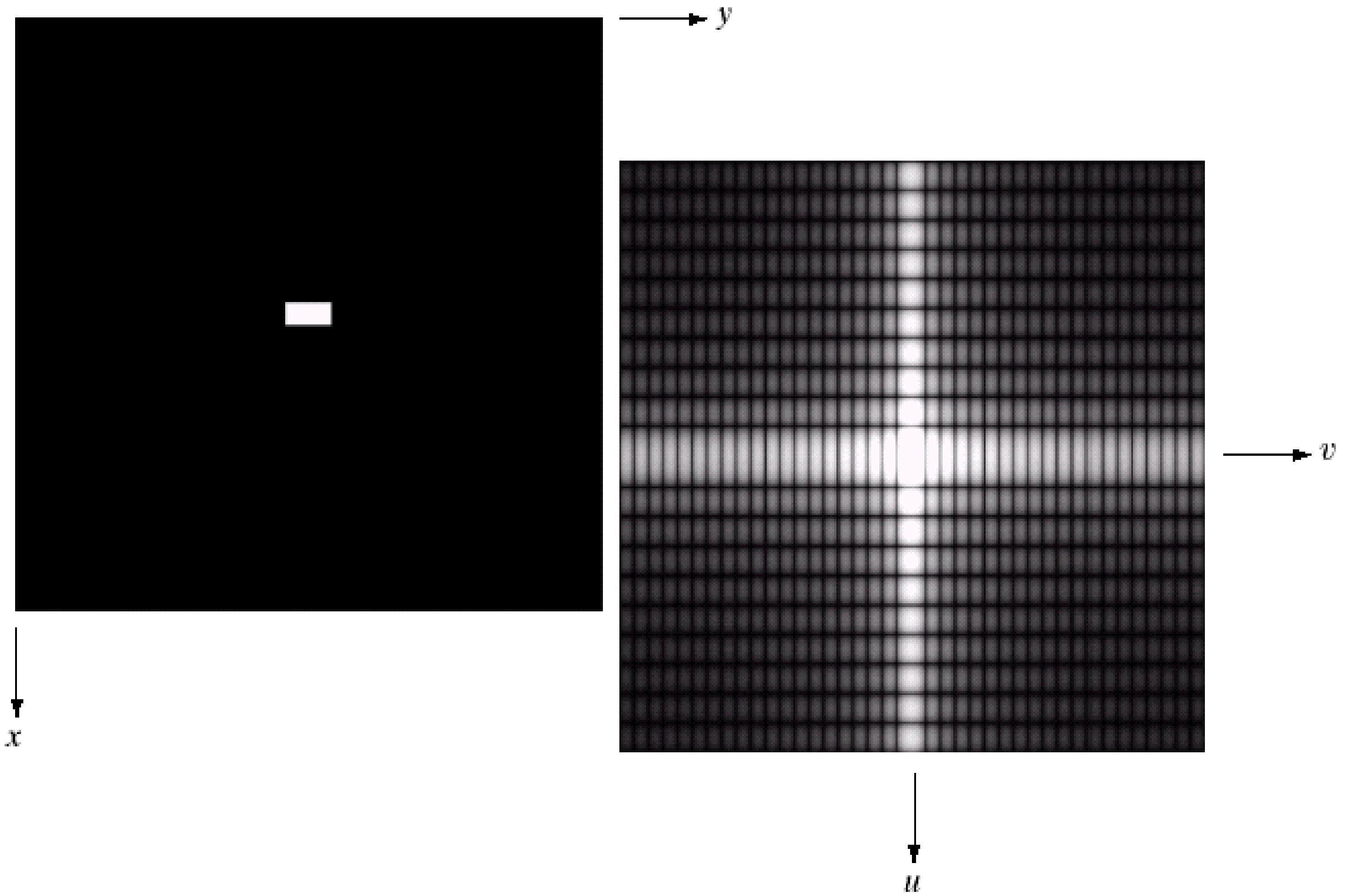
2-D FFT Shift (cont.) : How it works



Example of 2-D DFT

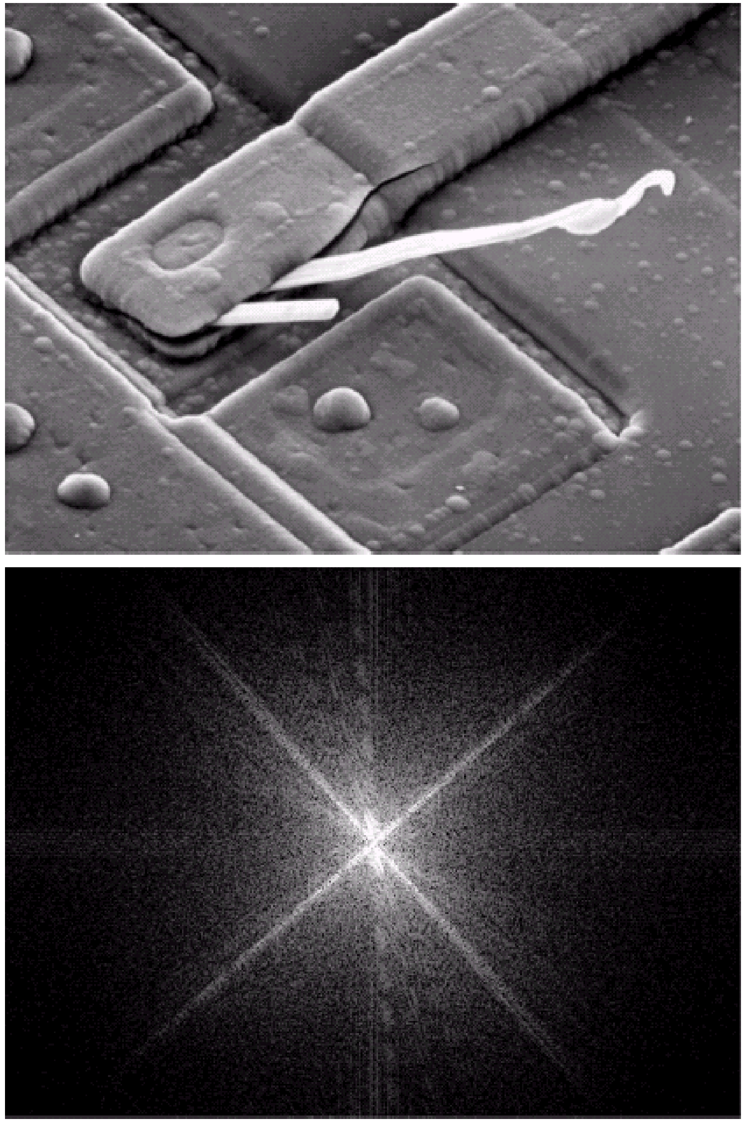
a b

FIGURE 4.3 (a) Image of a 20×40 white rectangle on a black background of size 512×512 pixels. (b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.



Notice that the longer the time domain signal,
The shorter its Fourier transform

Example of 2-D DFT

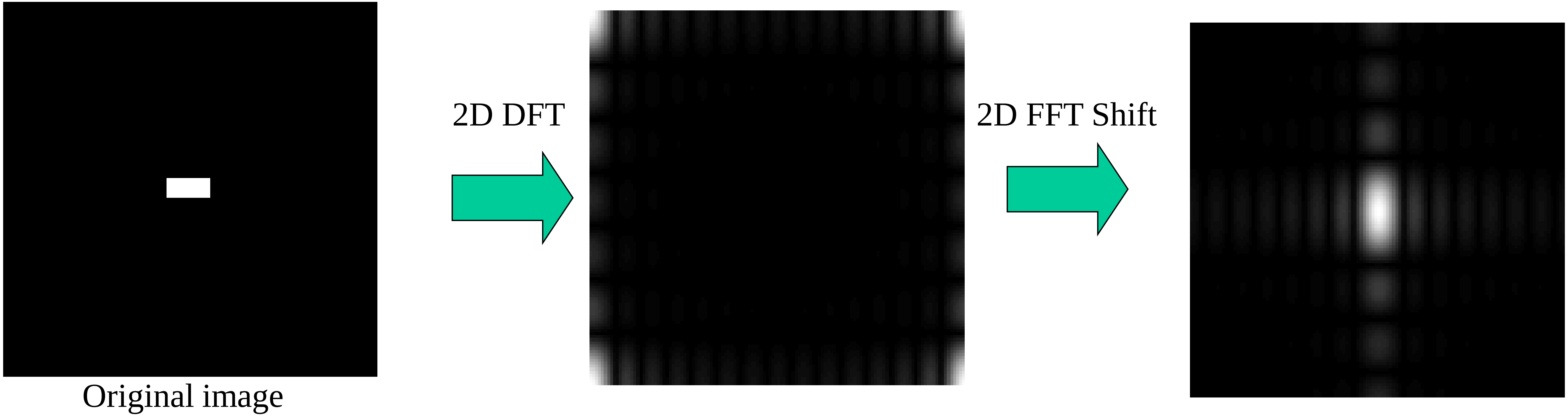


a
b

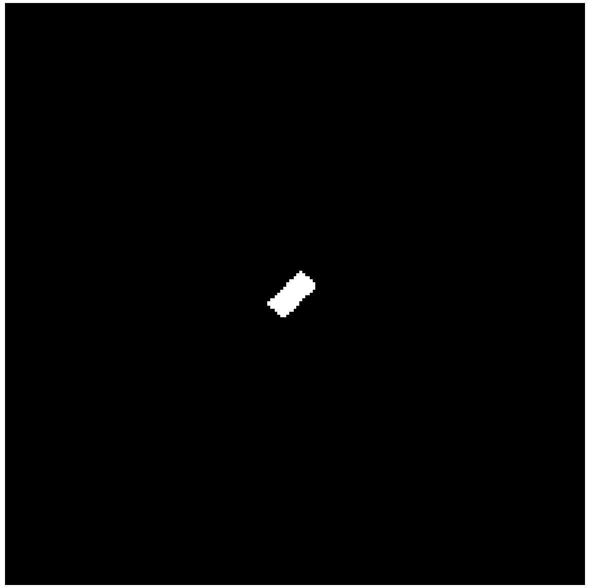
FIGURE 4.4 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

Notice that direction of an object in spatial image and Its Fourier transform are orthogonal to each other.

Example of 2-D DFT

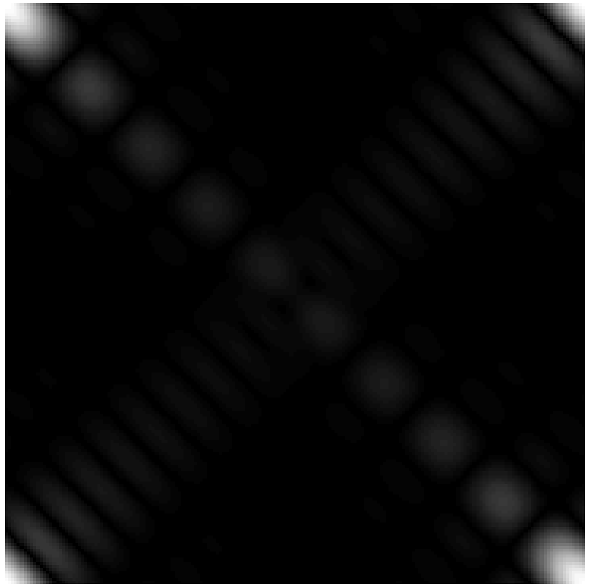


Example of 2-D DFT

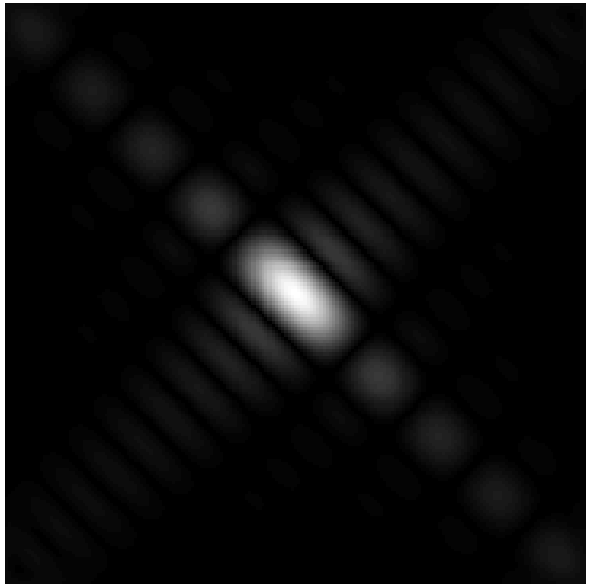


Original image

2D DFT



2D FFT Shift

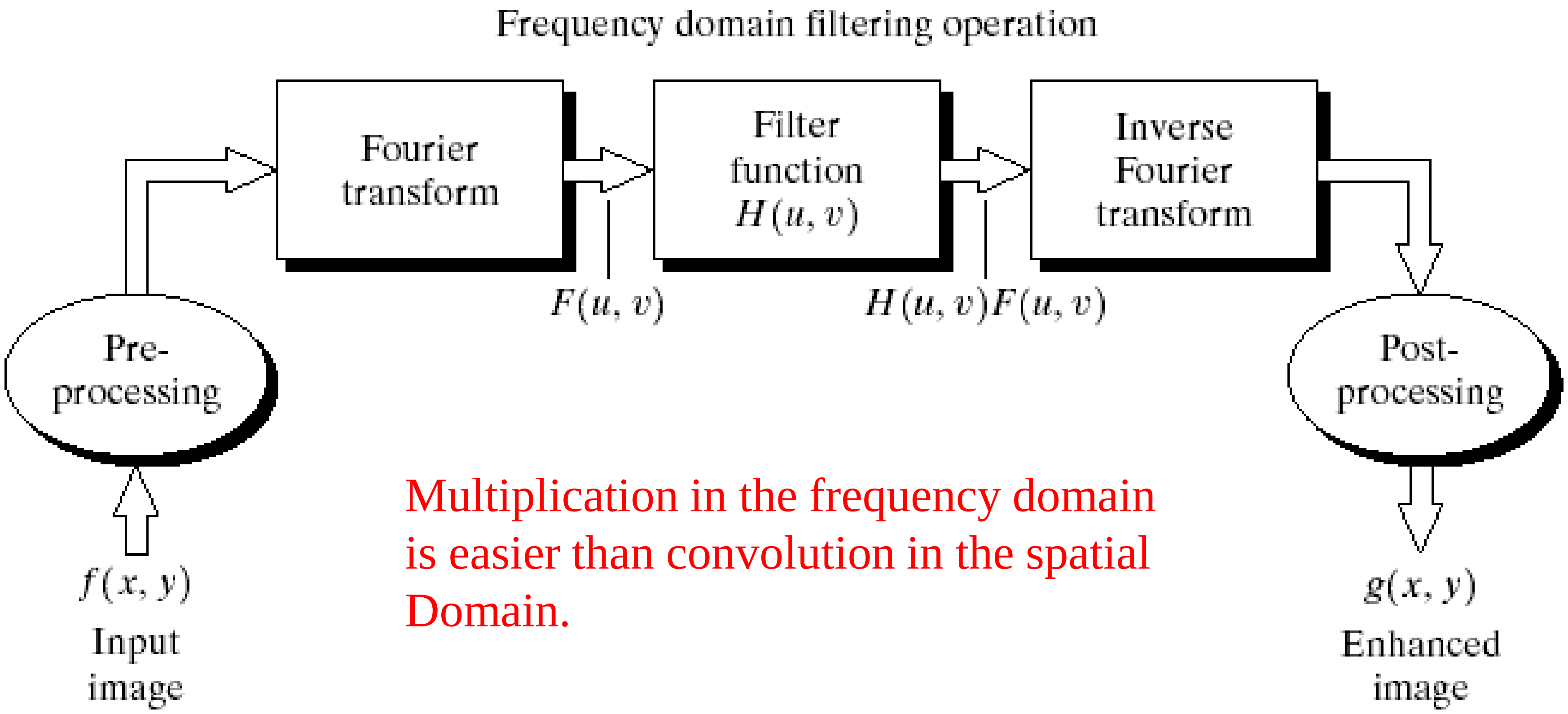


Basic Concept of Filtering in the Frequency Domain

- From Fourier Transform Property:

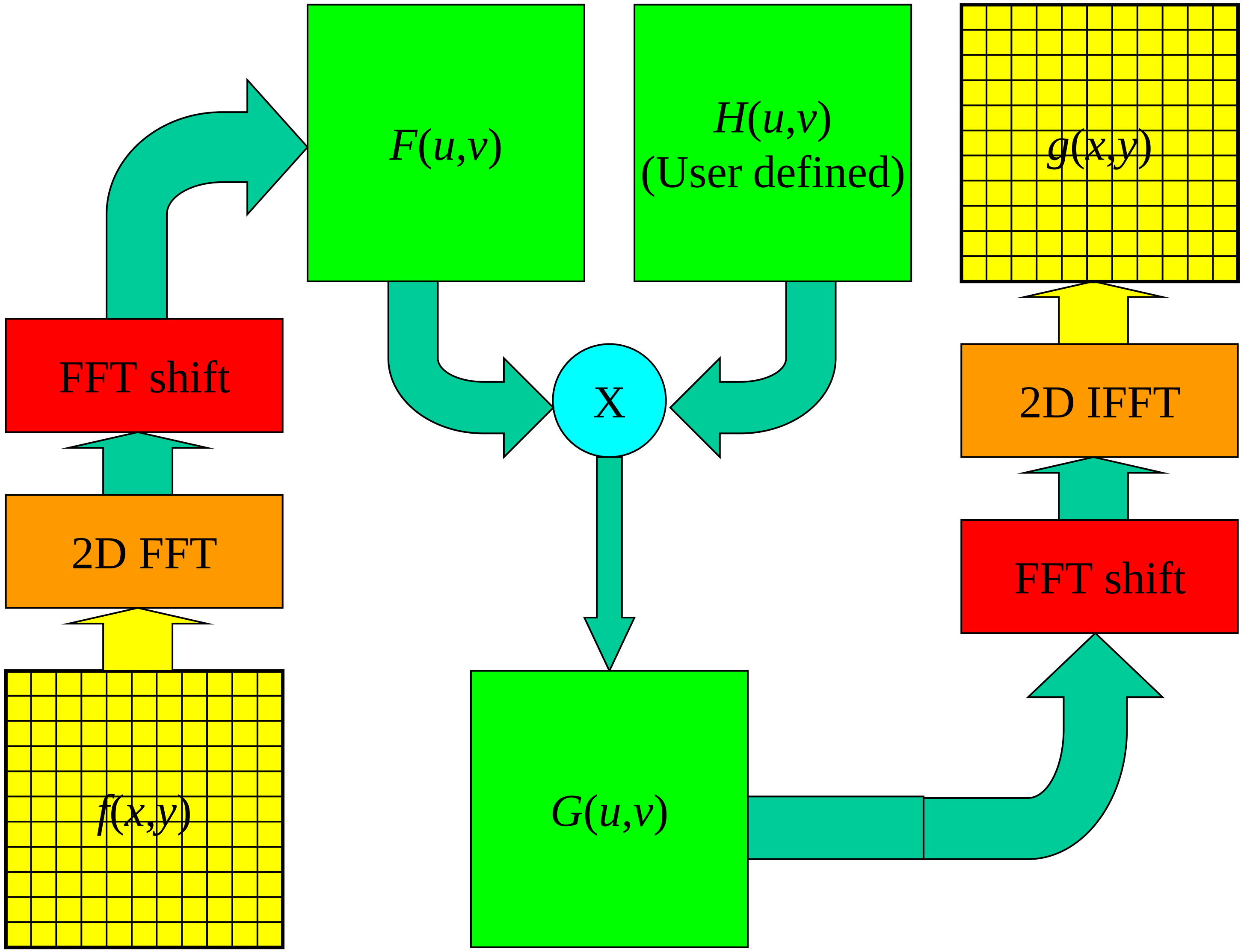
$$g(x,y) = f(x,y) * h(x,y) \Leftrightarrow F(u,v) \cdot H(u,v) = G(u,v)$$

We can perform filtering process by using



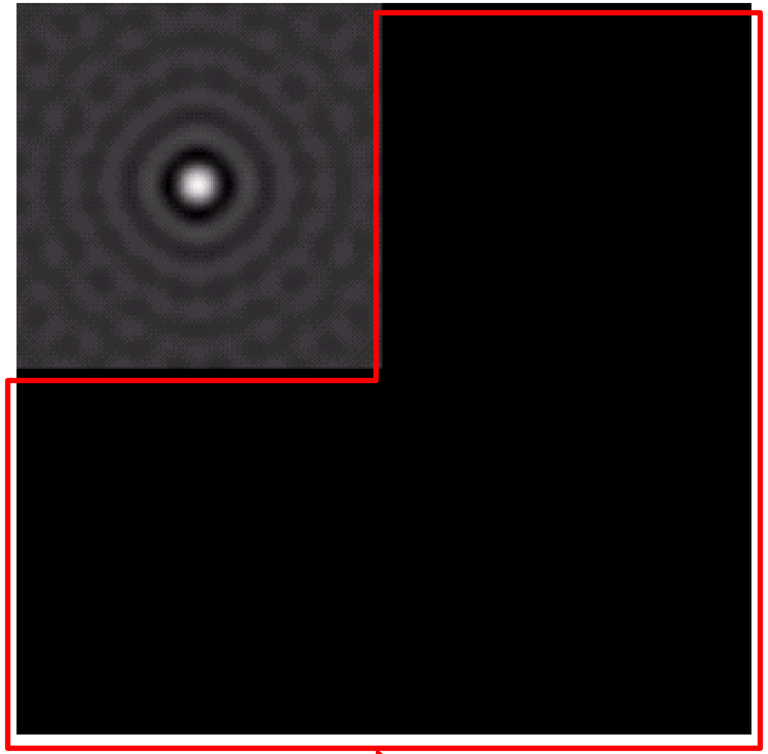
Multiplication in the frequency domain is easier than convolution in the spatial Domain.

Filtering in the Frequency Domain with FFT shift



In this case, $F(u,v)$ and $H(u,v)$ must have the same size and have the zero frequency at the center.

Linear Convolution by using Circular Convolution and Zero Padding



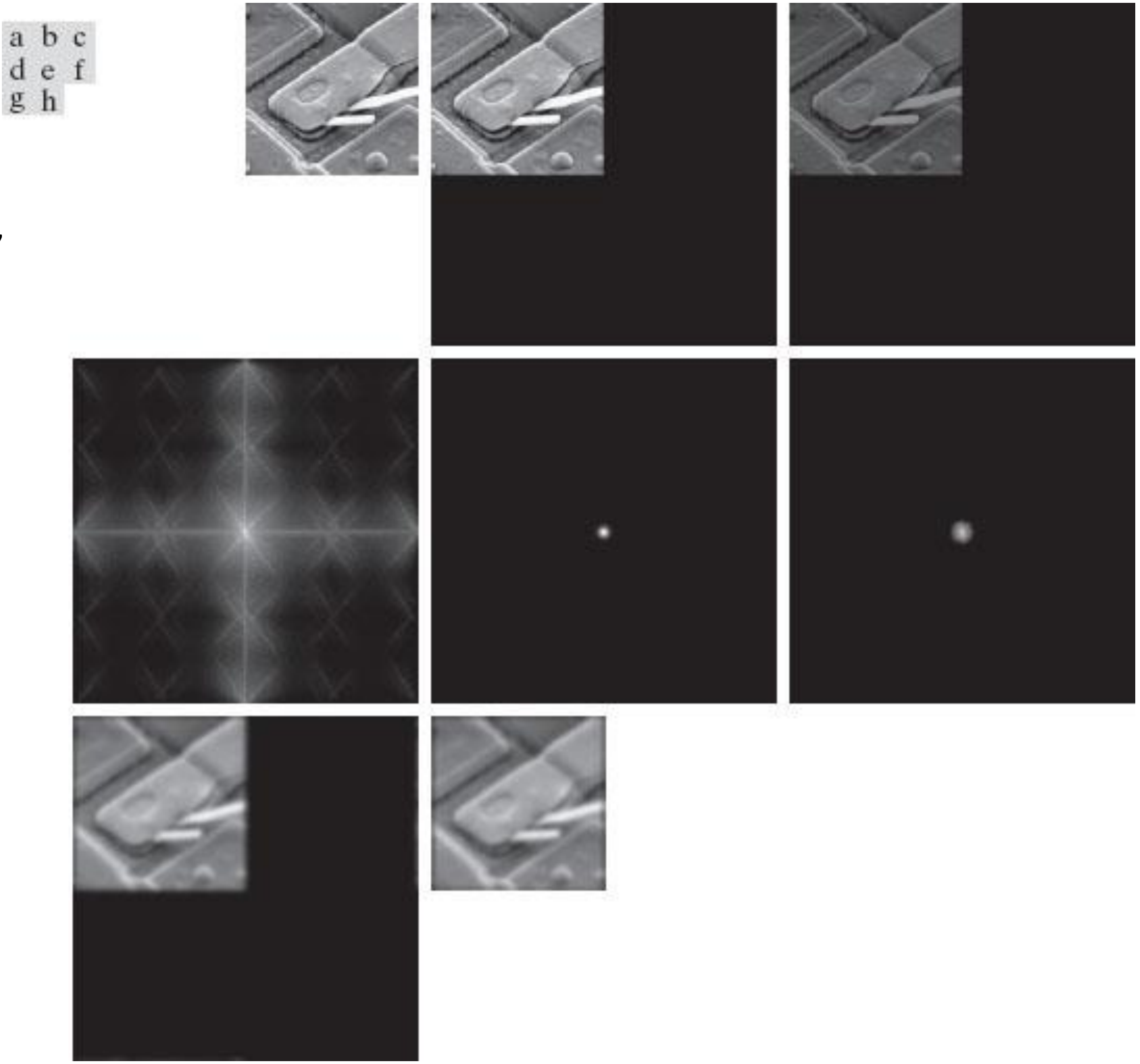
Zero padding area in the spatial
Domain of the mask image
(the ideal lowpass filter)

Filtered image

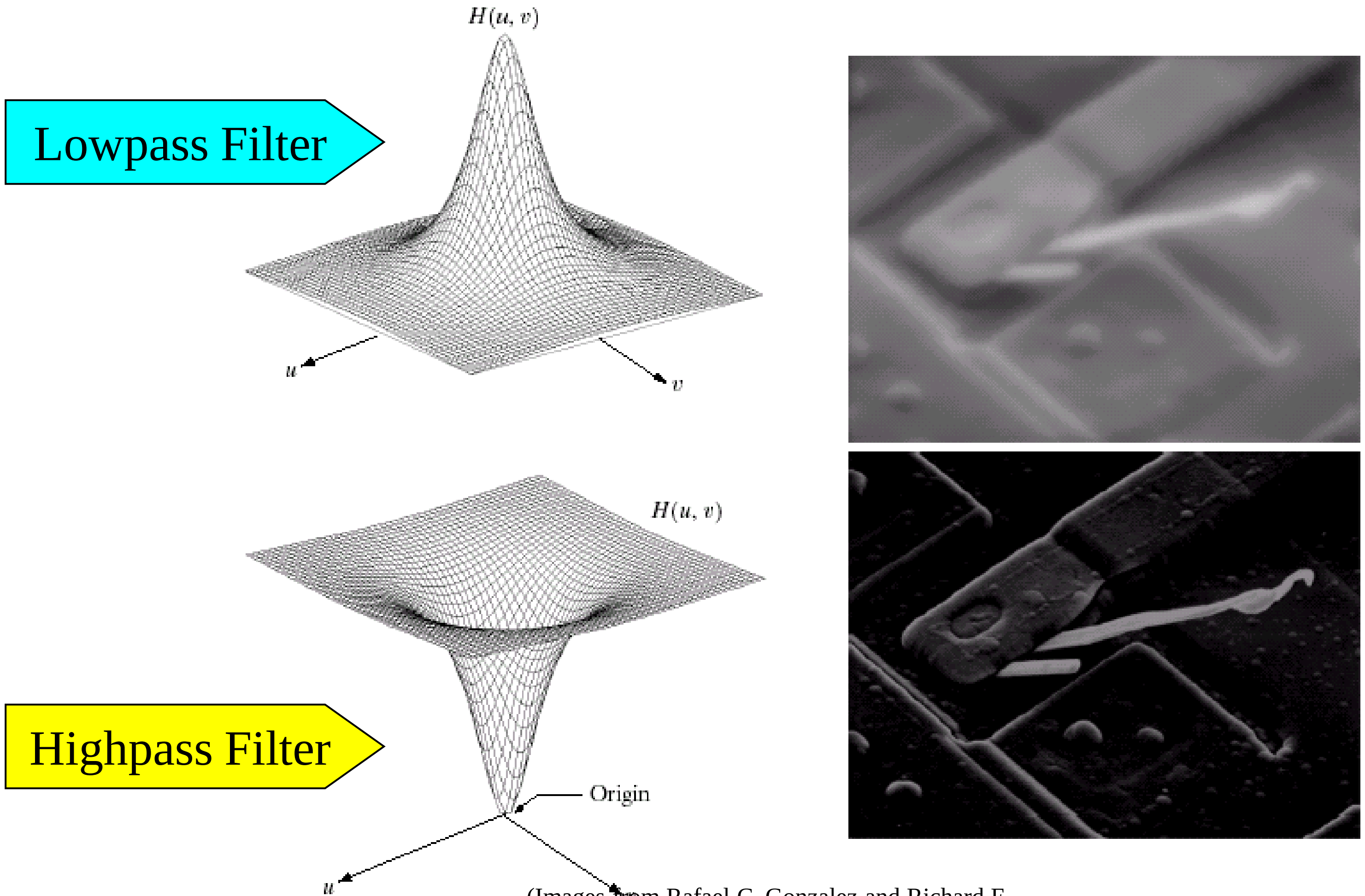
Only this area is kept.

Filtering by zero-padding

- (a) M x N image, f
- (b) padded image f_p of size P x Q
- (c) Result of multiplying f_p by $(-1)^{x+y}$
- (d) Spectrum of F
- (e) Centered Gaussian lowpass filter transfer function, H, of size P x Q
- (f) Spectrum of the product HF
- (g) Image g_p, the real part of the IDFT of HF, multiplied by $(-1)^{x+y}$
- (h) Final result obtained by extracting the first M rows and N columns of g_p



Filtering in the Frequency Domain : Example



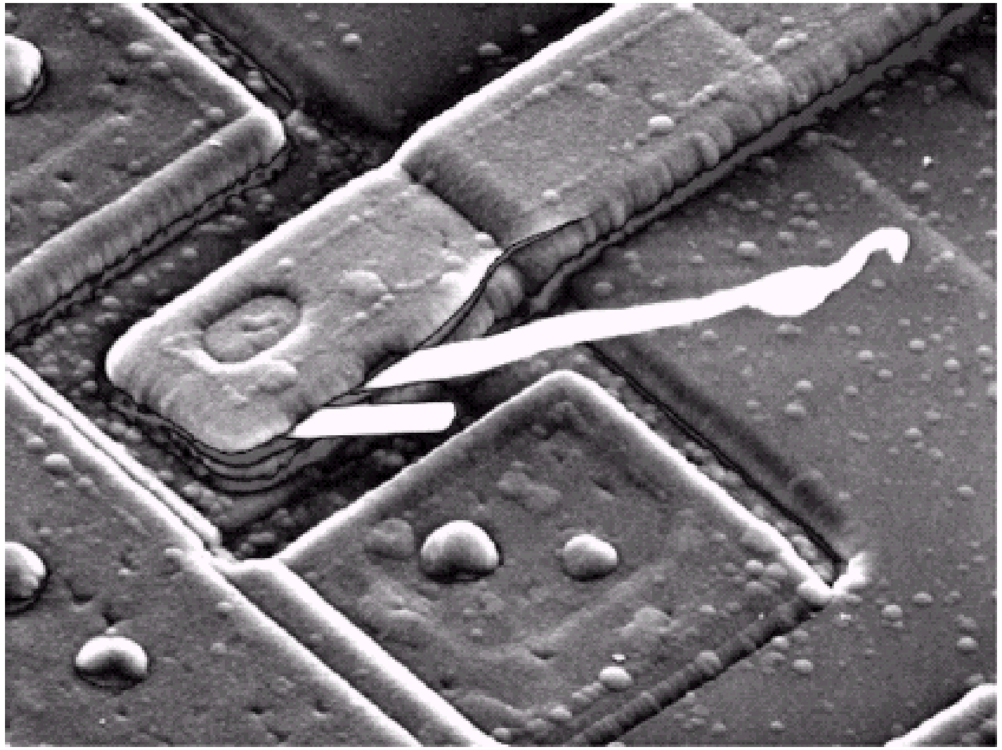
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

a b
c d

FIGURE 4.7 (a) A two-dimensional lowpass filter function. (b) Result of lowpass filtering the image in Fig. 4.4(a). (c) A two-dimensional highpass filter function. (d) Result of highpass filtering the image in Fig. 4.4(a).

Filtering in the Frequency Domain : Example (cont.)

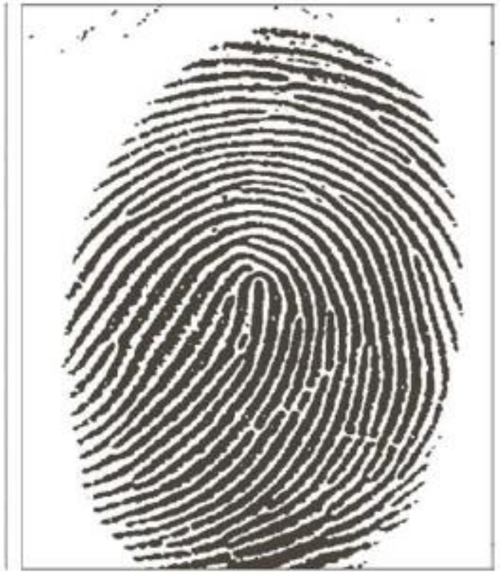
FIGURE 4.8
Result of highpass filtering the image in Fig. 4.4(a) with the filter in Fig. 4.7(c), modified by adding a constant of one-half the filter height to the filter function. Compare with Fig. 4.4(a).



Result of Sharpening Filter

Image Thresholding

- What is image/video segmentation?
  - Process of partitioning a digital image into multiple regions
  - Application
    - Object classification



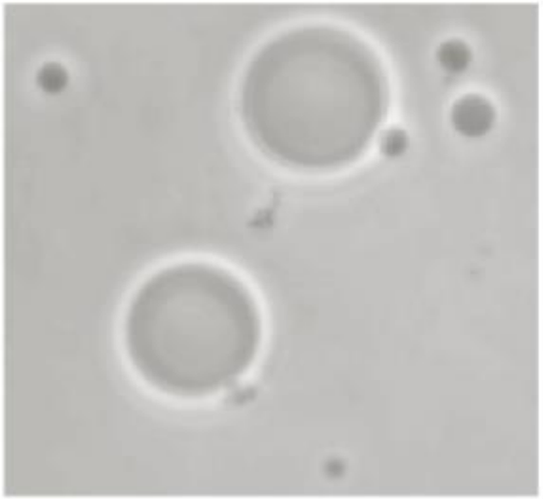
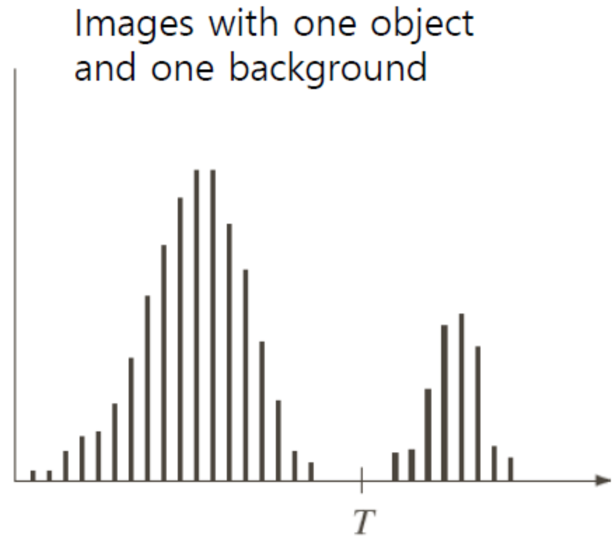


Image Thresholding

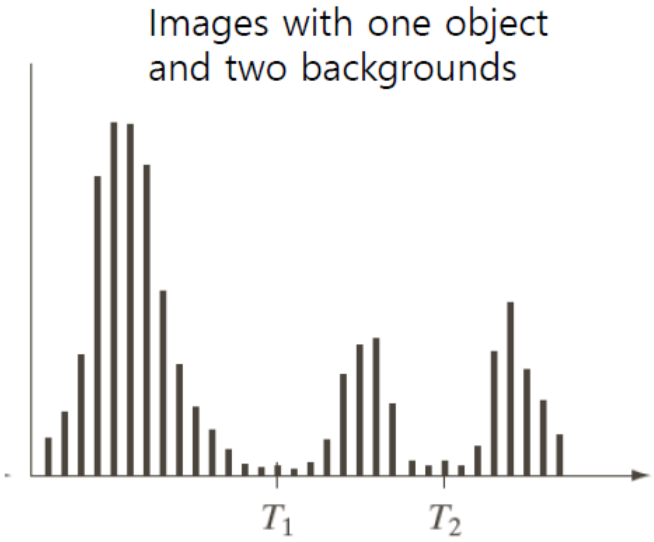
- What is image/video segmentation?
  - Input images are assumed to be gray-scale
    - Input: gray-scale image
    - Output: binary image (images with 0 and 255 (or 0 and 1) only)

Image Thresholding

- Basic concepts
  - Intensity of background and object is different
  - Background and object are homogenous



$$g(x,y)=\begin{cases}1 & if\ f(x,y)>T\\ 0 & otherwise\end{cases}$$

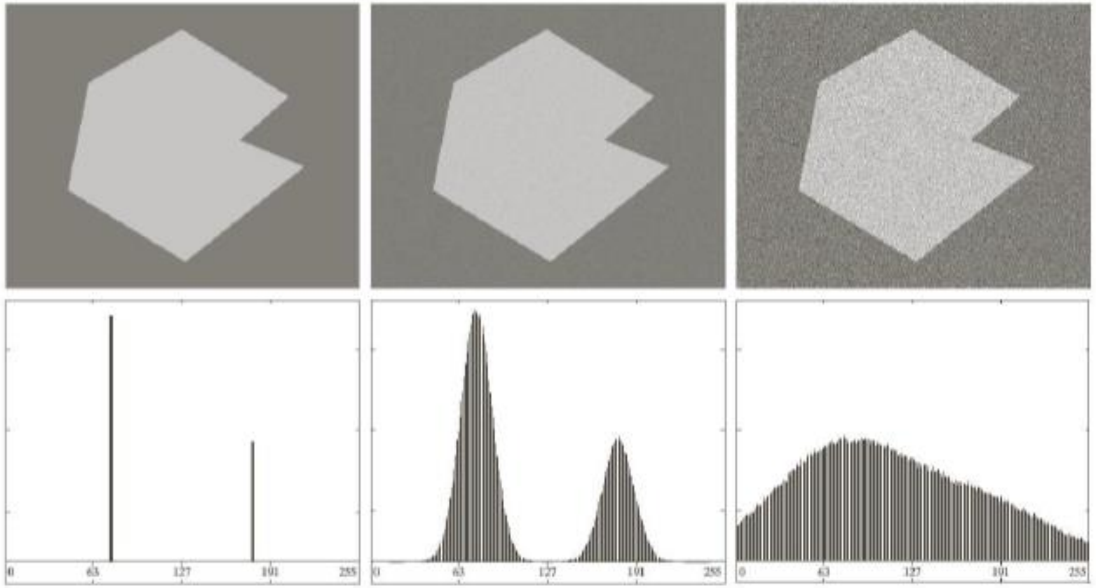


$$g(x,y)=\begin{cases}a & if\ f(x,y)>T_2\\ b & if\ T_1<f(x,y)\le T_2\\ c & otherwise\end{cases}$$

  - Finding the proper threshold is important

Image Thresholding

- Noise & Illumination, Reflectance



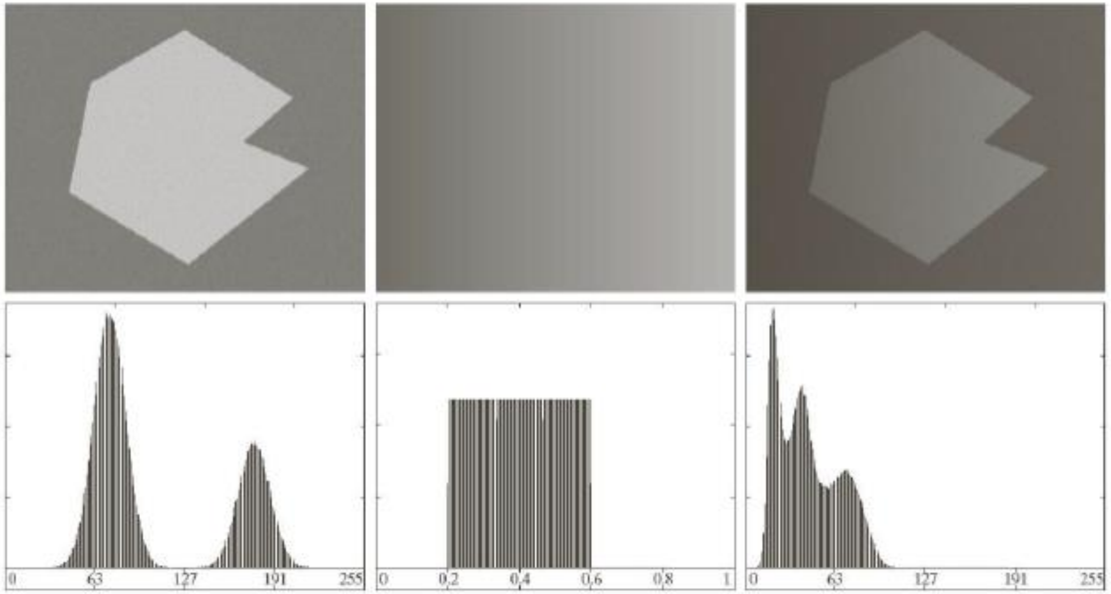


Image Thresholding

- Thresholding after applying smoothing

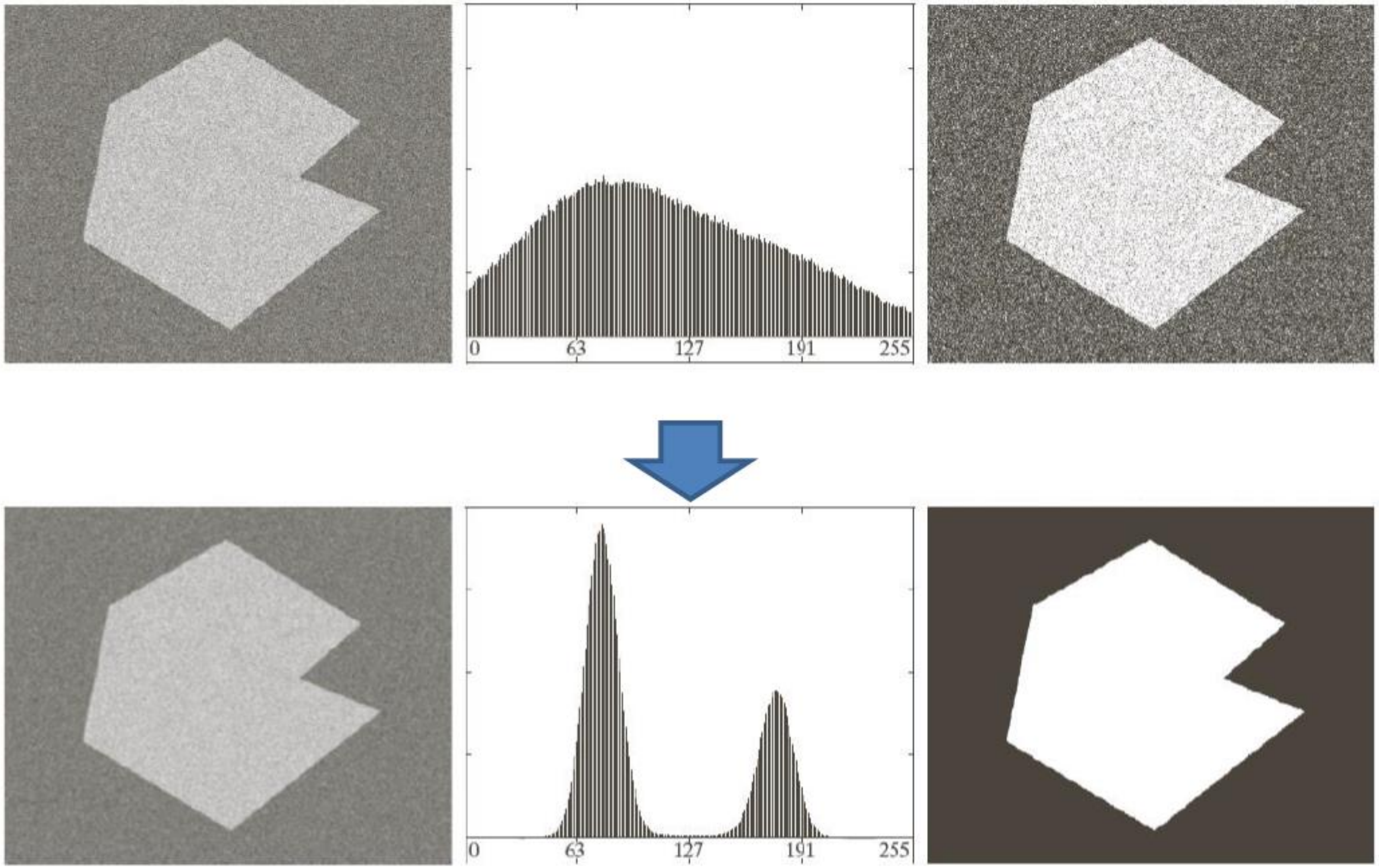


Image Thresholding

- Global thresholding
  - Use same threshold for every pixel

- Local (adaptive) thresholding
  - Use different threshold for each pixel

Global Thresholding

- Basic method

  1.Select an initial estimate for the global threshold T

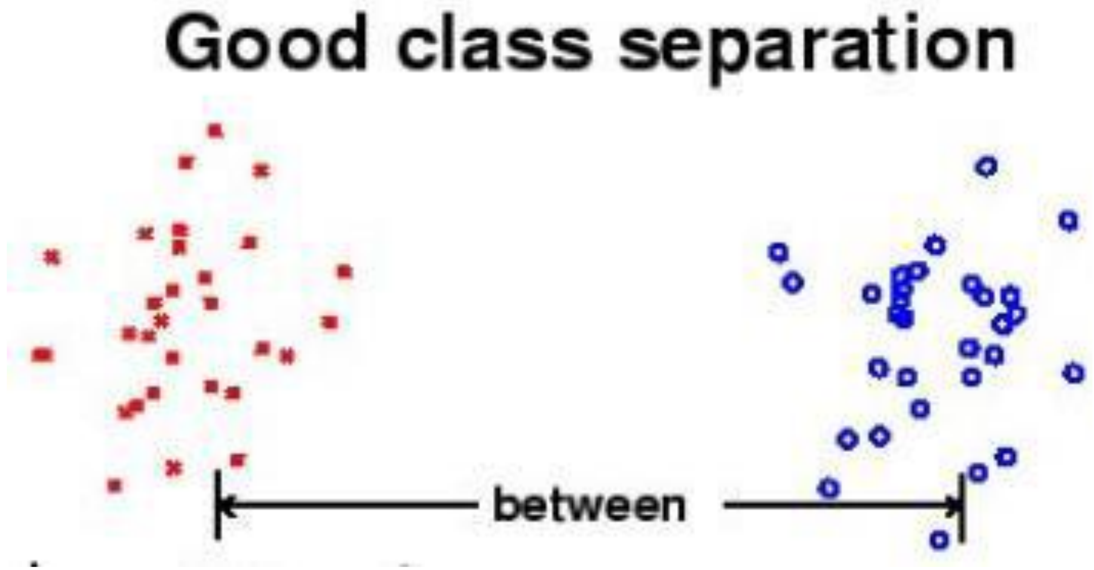
  2. Segment the image using T into two groups

  3. compute the mean(m1,m2) for each group

  4. compute new threshold as T=0.5X(m1+m2)

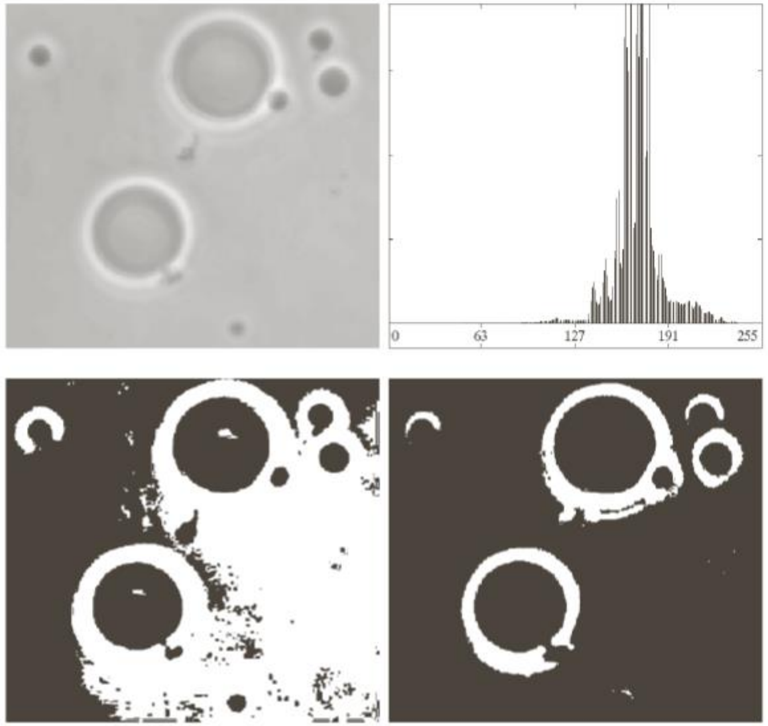
  5. repeat step 2 through 4 until the difference between values of T in successive iterations is small

- Otsu’s method
  - Concept
    - Well-thresholdedclasses should be distinct with respect to the intensity values of their pixels
    - Conversely, a threshold giving the best separation between classes would be the best threshold
    - It is based on computations performed on the histogram of an image



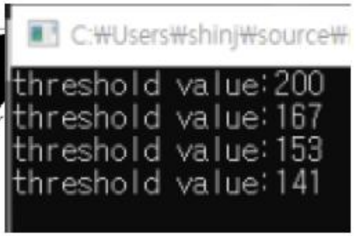
Global Thresholding

- Otsu's method
  1. Compute the normalized histogram
  2. For each threshold k, compute between-class variance $\sigma^2{}_B$
  3. Obtain the Otsu threshold k for which $\sigma^2{}_B$ is maximized



Global Thresholding

- Basic method



- Otsu’s method

Local(Adaptive) Thresholding

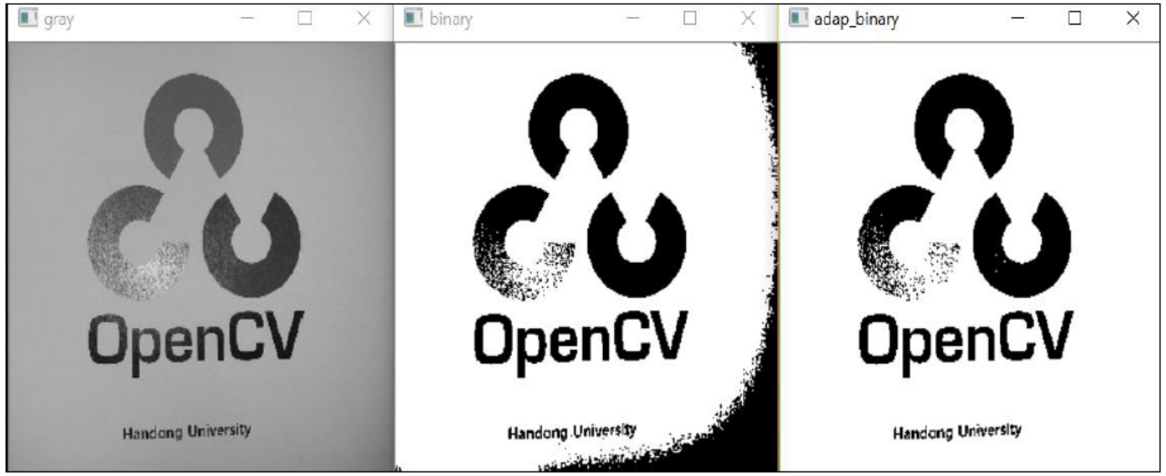
- Set a threshold for each point depending on the intensity distributions of adjacent pixels

ADAPTIVE_THRESH_MEAN_C :

$$T(x,y) = mean\ of\ the\ blocksize \times blocksize\ neighborhood\ of\ (x,y) - C$$

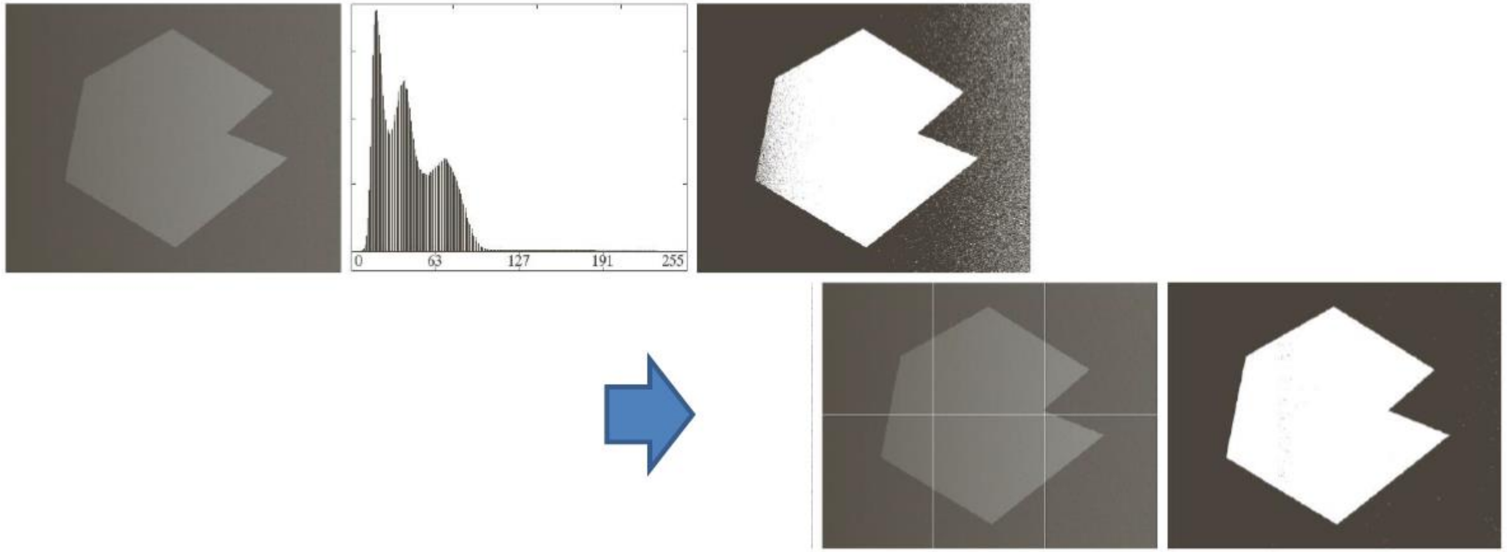
ADAPTIVE_THRESH_GAUSSIAN_C :

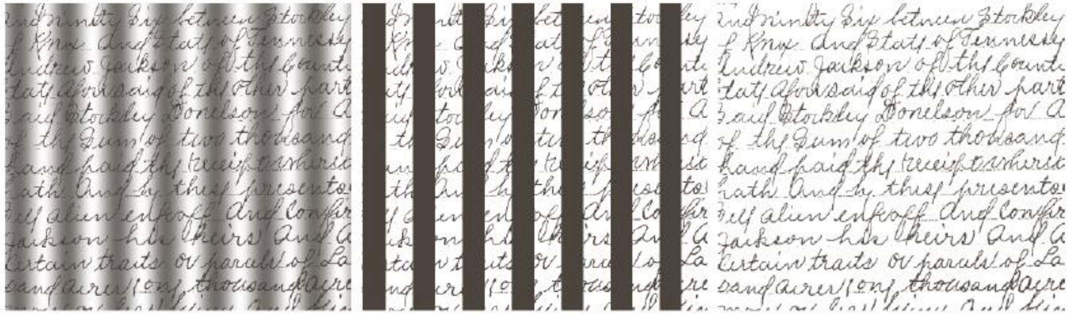
$$T(x,y)$$
$$= a\ weighted\ sum(cross - correlation\ with\ a\ Gaussian\ windonw)\ of\ the\ blocksize$$
$$\times\ blocksize\ neighborhood\ of\ (x,y) - C$$



Local(Adaptive) Thresholding

- Set a threshold for each point depending on the intensity distributions of adjacent pixels
  - Image partitioning





9.2.1 Erosion

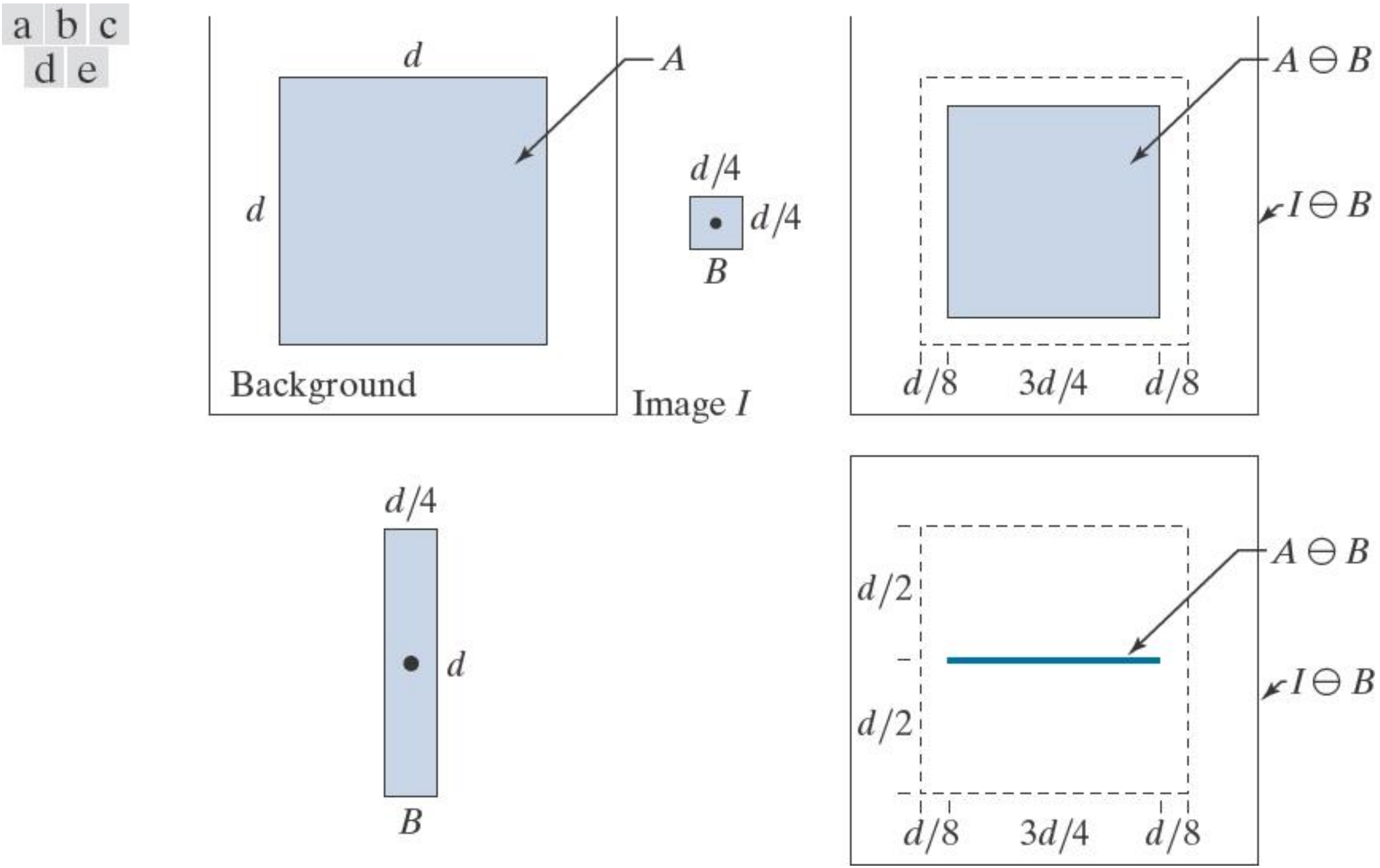
- Erosion is used for shrinking of element A by using element B
- Erosion for Sets A and B in Z2, is defined by the following equation:

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \qquad (9.2\text{-}1)$$

$$A \ominus B = \{z \mid (B)_z \cap A^c = \varnothing\} \qquad (9.2\text{-}2)$$

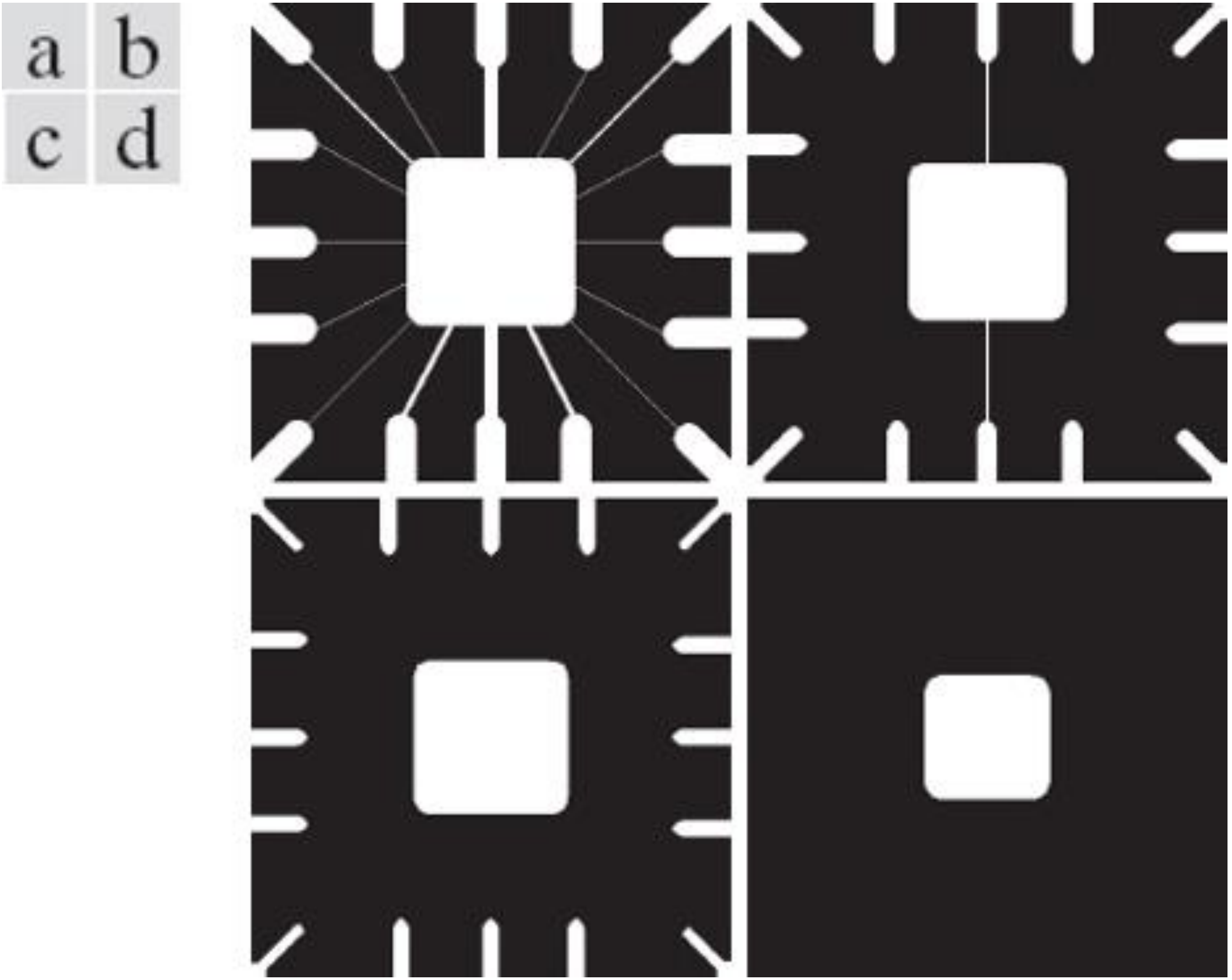
- This equation indicates that the erosion of A by B is the set of all points z such that B, translated by z, is contained in A.
- Erosion can be used to
  - Shrinks or thins objects in binary images
  - Remove image components(how?)
    - Erosion is a morphological filtering operation in which image details smaller than the structuring elements are filtered(removed)

9.2.1 Erosion – Example



9.2.1 Erosion – Example

- (a) 486 x 486 binary image
- (b)-(d) square structuring elements of sizes 11x11, 15x15, 45x45

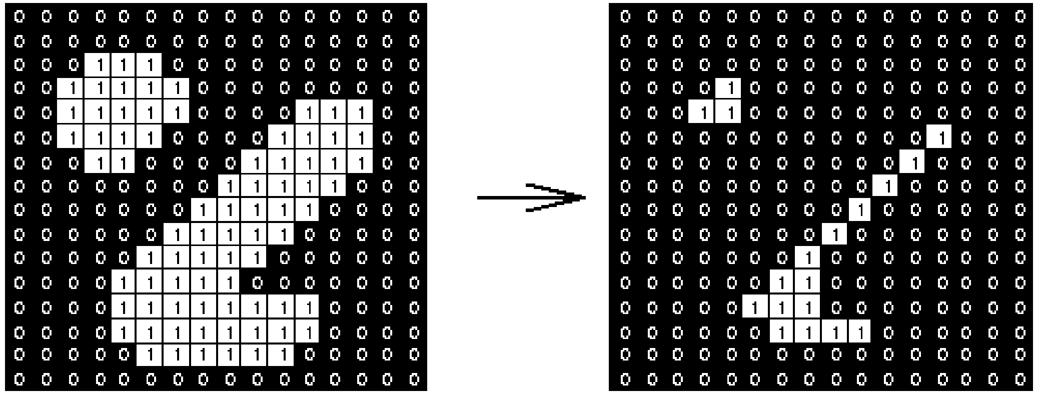


Erosion

- Suppose that the structuring element is a 3×3 square
- Note that in subsequent diagrams, foreground pixels are represented by 1's and background pixels by 0's.
- The structuring element is now superimposed over each foreground pixel (input pixel) in the image. If all the pixels below the structuring element are foreground pixels then the input pixel retains it’s value. But if any of the pixels is a background pixel then the input pixel gets the background pixel value.

1	1	1
1	1	1
1	1	1

Structuring element



9.2.2 Dilation

- Dilation is used for **expanding an element A by using structuring element B**
- Dilation of A by B and is defined by the following equation:

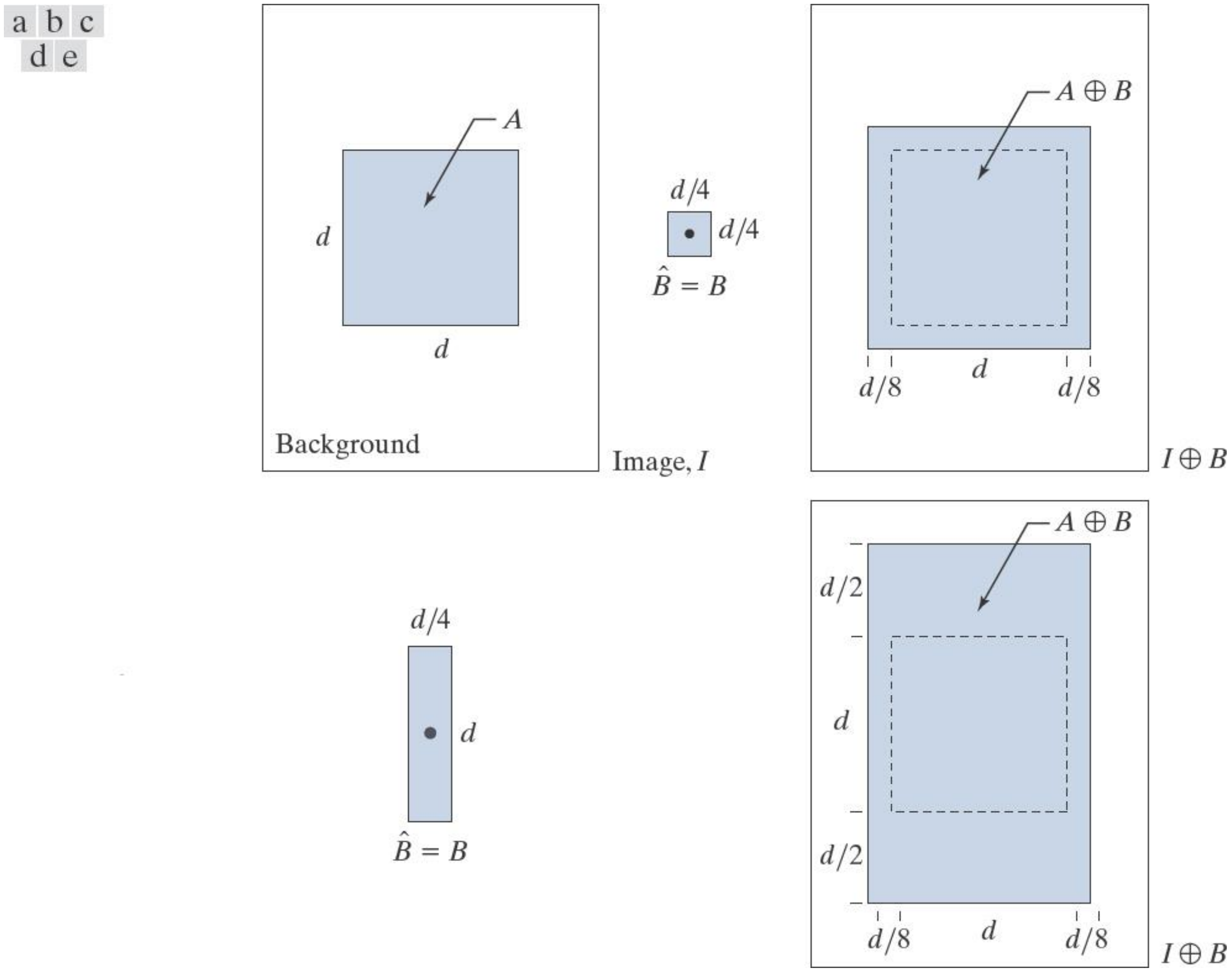
$$A \oplus B = \left\{ z \mid (\hat{B})_z \cap A \neq \varnothing \right\} \qquad (9.2\text{-}3)$$

- This equation is based on obtaining the reflection of B about its origin and shifting this reflection by z.
- The dilation of A by B is the set of all displacements z, such that $\hat{B}$ and A overlap b y at least one element. Based on this interpretation the equation of (9.2-1) can be rewritten as:

$$A \oplus B = \left\{ z \mid [(\hat{B})_z \cap A] \subseteq A \right\} \qquad (9.2\text{-}4)$$

  - Relation to Convolution mask:
    - Flipping
    - Overlapping

9.2.2 Dilation – Example

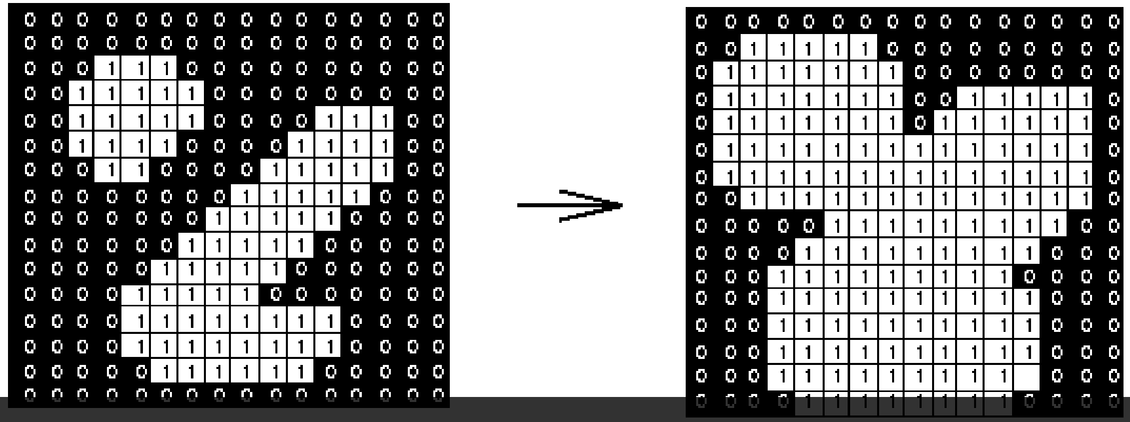


Dilation

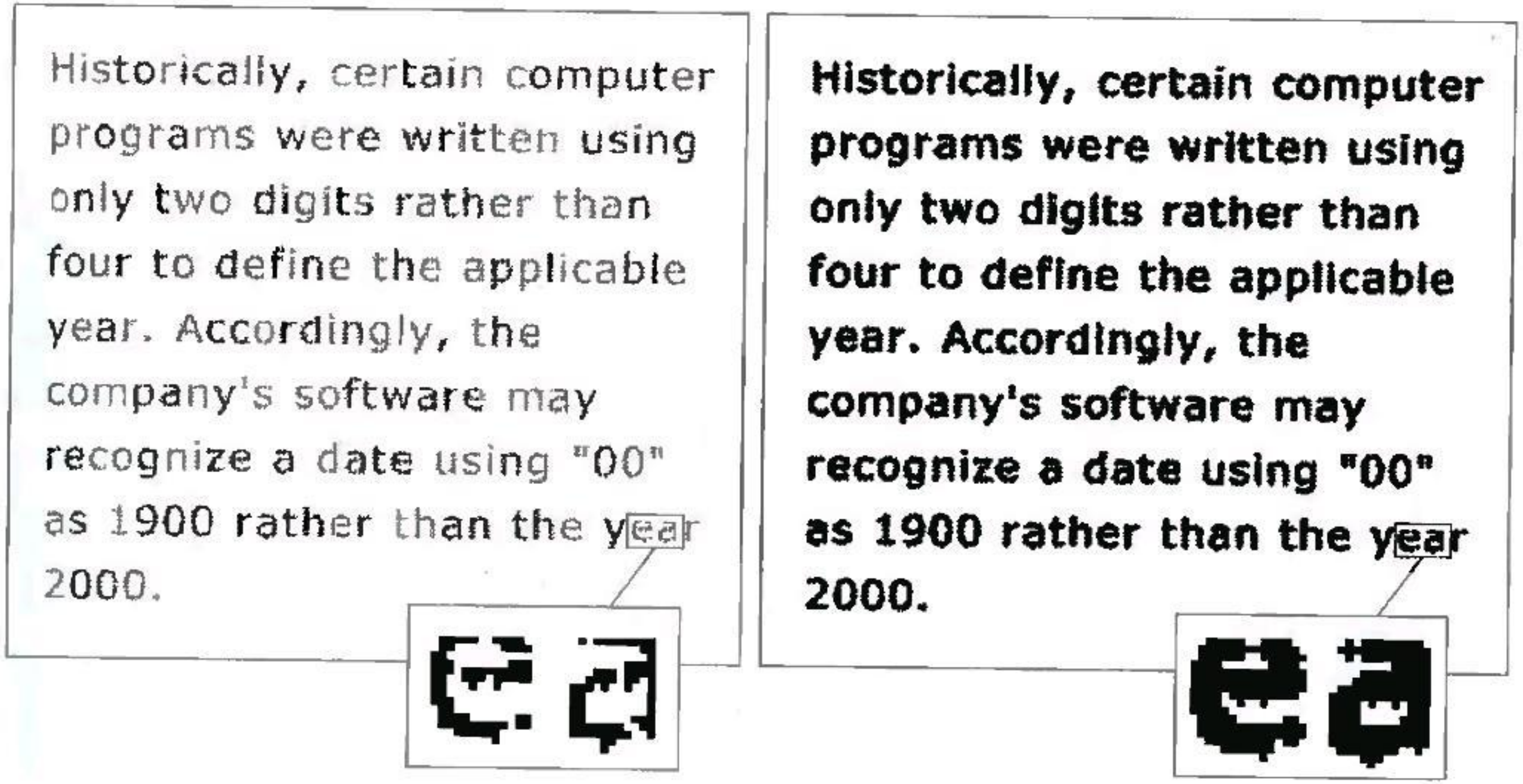
- Suppose that the structuring element is a 3×3 square .
- Note that in subsequent diagrams, foreground pixels are represented by 1's and background pixels by 0's.
- To compute the dilation of a binary input image by this structuring element, we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel position.
- If the center pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value.

1	1	1
1	1	1
1	1	1

Structuring element



9.2.2 Dilation – A More interesting Example (bridging gaps)

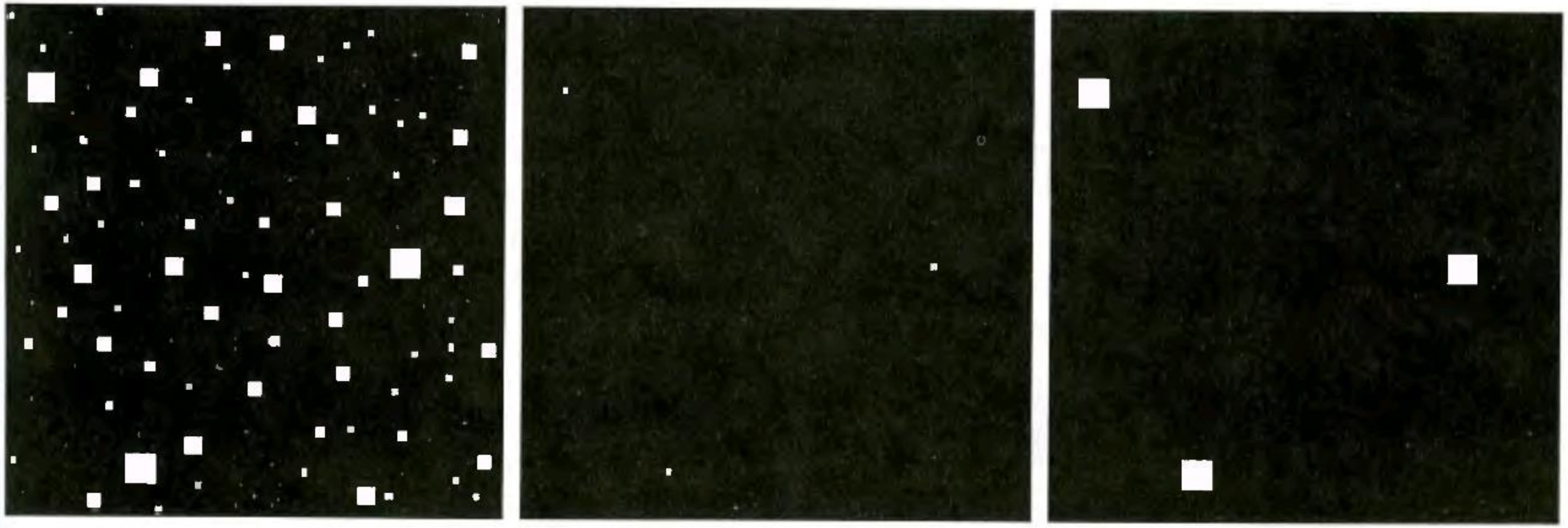


a c
b

FIGURE 9.5
(a) Sample text of poor resolution with broken characters (magnified view). (b) Structuring element. (c) Dilation of (a) by (b). Broken segments were joined.

0	1	0
1	1	1
0	1	0

Erosion and Dilation summary



a b c

FIGURE 9.7 (a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

Erosion vs. Dilation

- Erosion:
  - **Shrinks** or **thins** objects in binary images
  - Remove image components(how?)
  - Erode away the boundaries of regions of foreground pixels
  - Areas of foreground pixels shrink in size, and holes within those areas become larger
- Dilation:
  - **Grows** or **thickens** object in a binary image
  - Bridging gaps
  - Fill small holes of sufficiently small size

9.3 Opening And Closing

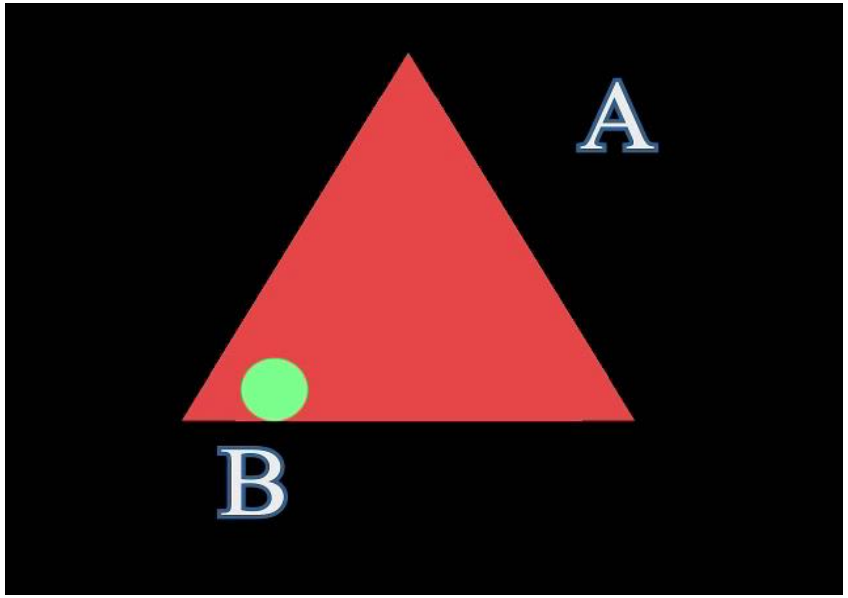
- **Opening** – smoothes contours , eliminates protrusions
- **Closing** – smoothes sections of contours, fuses narrow breaks and long thin gulfs, eliminates small holes and fills gaps in contours
- These operations are dual to each other
- These operations are can be applied few times, but has effect only once

9.3 Opening And Closing

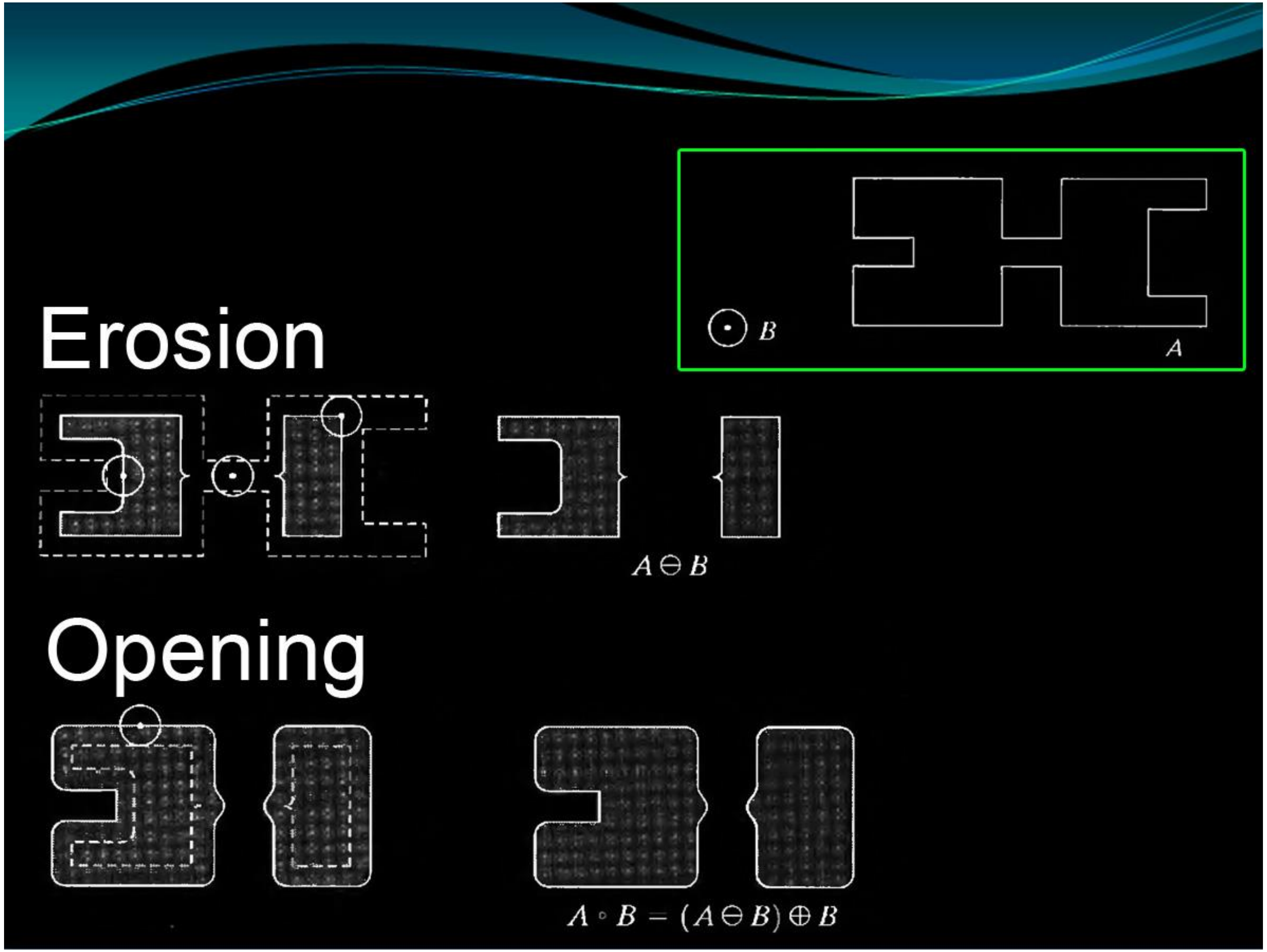
- Opening –
  - First – erode A by B, and then dilate the result by B
  - In other words, opening is the unification of all B objects Entirely Contained in A

-

$$A \circ B = (A \ominus B) \oplus B$$

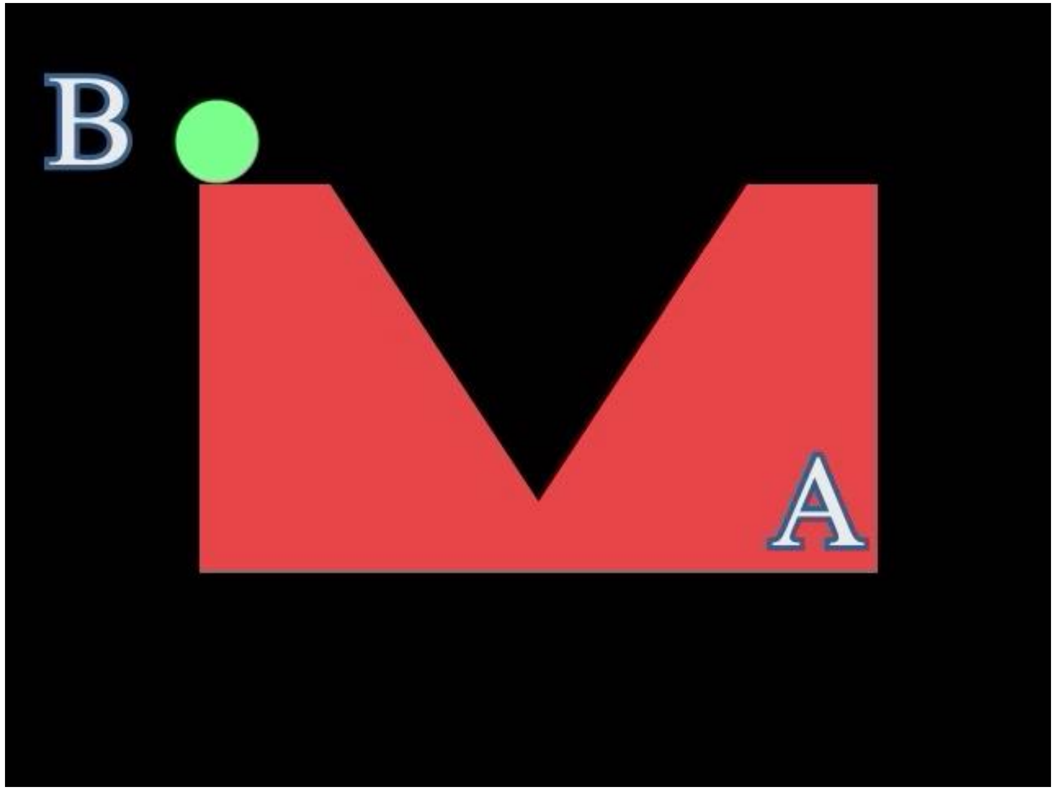


Erosion vs. Opening

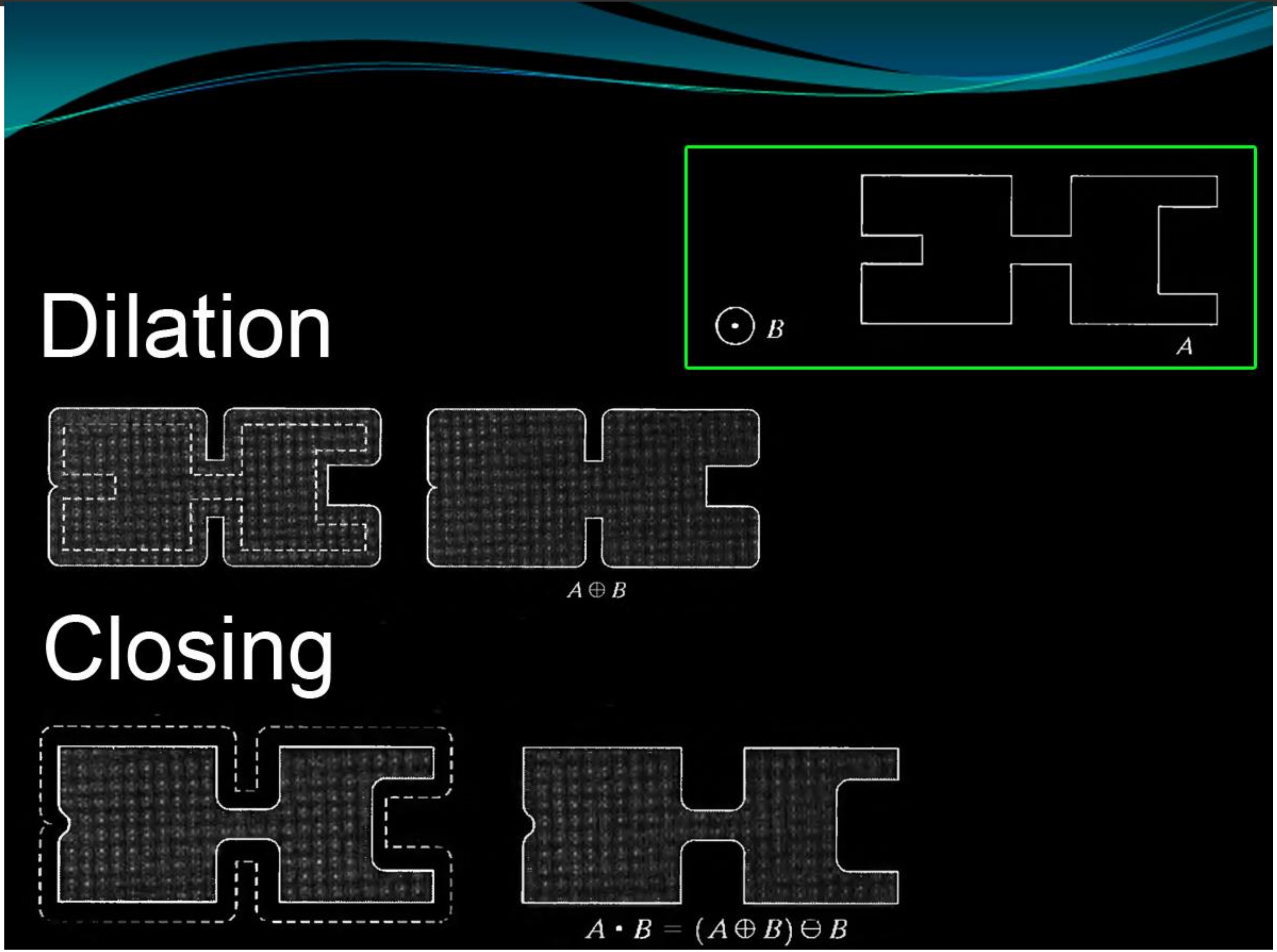


- Closing –
  - First – dilate A by B, and then erode the result by B
  - In other words, closing is the group of points, which the intersection of object B around them with object A – is not empty

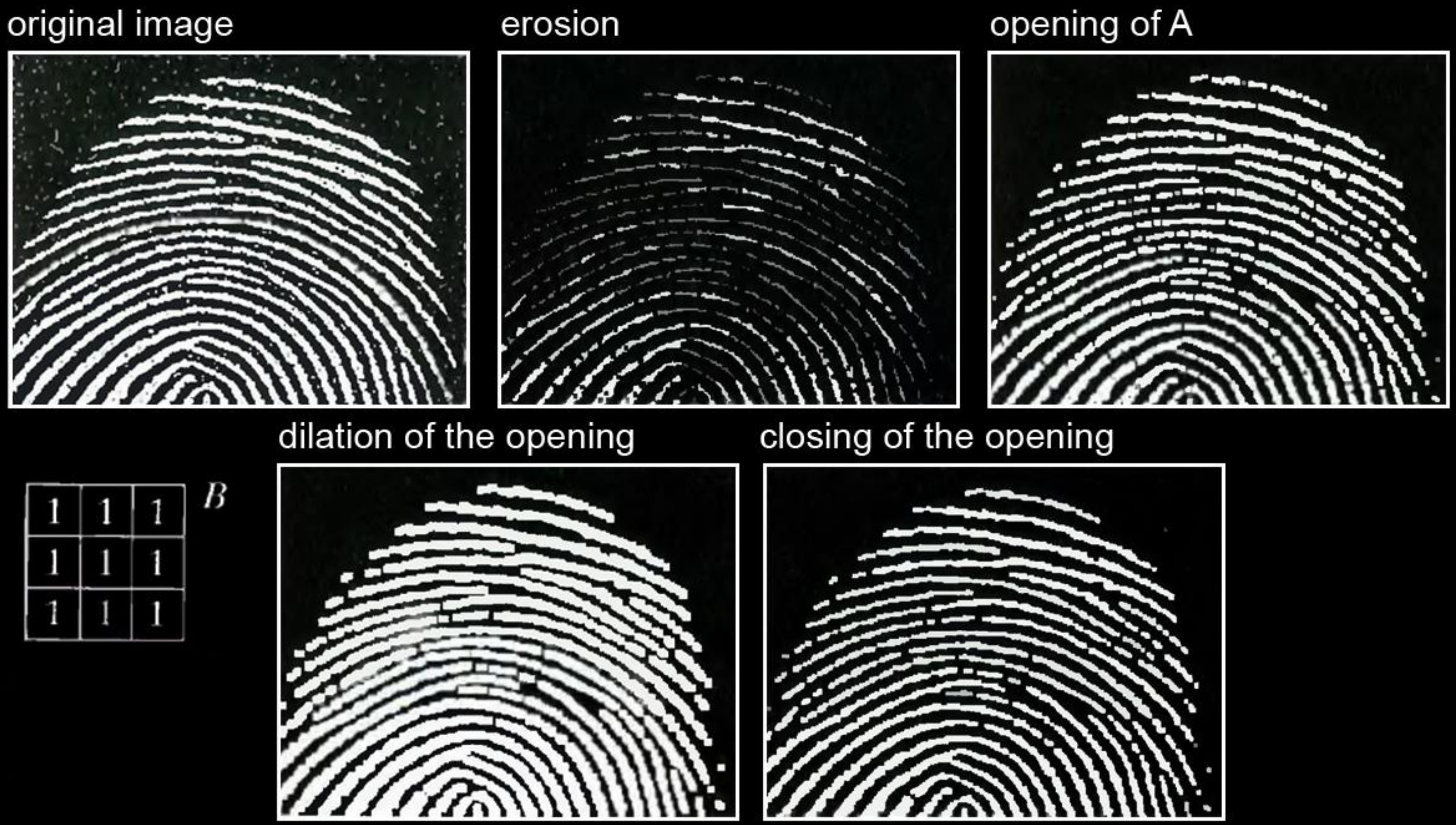
$$A \cdot B = (A \oplus B) \ominus B$$



Dilation vs. Closing



Use of opening and closing for morphological filtering



Computing gradient image using Sobel filter

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

image = cv2.imread('../data/Lena.png', 0)

dx = cv2.Sobel(image, cv2.CV_32F, 1, 0)
dy = cv2.Sobel(image, cv2.CV_32F, 0, 1)

plt.figure(figsize=(8,3))
plt.subplot(131)
plt.axis('off')
plt.title('image')
plt.imshow(image, cmap='gray')
plt.subplot(132)
plt.axis('off')
plt.imshow(dx, cmap='gray')
plt.title(r'$\frac{dI}{dx}$')
plt.subplot(133)
plt.axis('off')
plt.title(r'$\frac{dI}{dy}$')
plt.imshow(dy, cmap='gray')
plt.tight_layout()
plt.show()
```

Image sharpening using Unsharp mask

```
import cv2
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt

image = cv2.imread('../data/Lena.png')

KSIZE = 11
ALPHA = 2
kernel = cv2.getGaussianKernel(KSIZE, 0)
kernel = -ALPHA * kernel @ kernel.T
kernel[KSIZE//2, KSIZE//2] += 1 + ALPHA
print(kernel.shape, kernel.dtype, kernel.sum())

filtered = cv2.filter2D(image, -1, kernel)

plt.figure(figsize=(8,4))
plt.subplot(121)
plt.axis('off')
plt.title('image')
plt.imshow(image[:, :, [2, 1, 0]])
plt.subplot(122)
plt.axis('off')
plt.title('filtered')
plt.imshow(filtered[:, :, [2, 1, 0]])
plt.tight_layout(True)
plt.show()

cv2.imshow('before', image)
cv2.imshow('after', filtered)
cv2.waitKey()
cv2.destroyAllWindows()
```

Image filtering using Gabor filter

```
import math
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('../data/Lena.png', 0).astype(np.float32) / 255

kernel = cv2.getGaborKernel((21, 21), 5, 1, 10, 1, 0, cv2.CV_32F)
kernel /= math.sqrt((kernel * kernel).sum())

filtered = cv2.filter2D(image, -1, kernel)

plt.figure(figsize=(8,3))
plt.subplot(131)
plt.axis('off')
plt.title('image')
plt.imshow(image, cmap='gray')
plt.subplot(132)
plt.title('kernel')
plt.imshow(kernel, cmap='gray')
plt.subplot(133)
plt.axis('off')
plt.title('filtered')
plt.imshow(filtered, cmap='gray')
plt.tight_layout()
plt.show()
```

Discrete Fourier Transform

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('../data/Lena.png', 0).astype(np.float32) / 255

fft = cv2.dft(image, flags=cv2.DFT_COMPLEX_OUTPUT)

shifted = np.fft.fftshift(fft, axes=[0, 1])
magnitude = cv2.magnitude(shifted[:,:,0], shifted[:,:,1])
magnitude = np.log(magnitude)

plt.axis('off')
plt.imshow(magnitude, cmap='gray')
plt.tight_layout(True)
plt.show()

restored = cv2.idft(fft, flags=cv2.DFT_SCALE | cv2.DFT_REAL_OUTPUT)

cv2.imshow('restored', restored)
cv2.waitKey()
cv2.destroyAllWindows()
```

Frequency-based Filtering

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

image = cv2.imread('../data/Lena.png', 0).astype(np.float32) / 255

fft = cv2.dft(image, flags=cv2.DFT_COMPLEX_OUTPUT)
fft_shift = np.fft.fftshift(fft, axes=[0, 1])
sz = 25
mask = np.zeros(fft.shape, np.uint8)
mask[image.shape[0]//2-sz:image.shape[0]//2+sz,
     image.shape[1]//2-sz:image.shape[1]//2+sz, :] = 1
fft_shift *= mask
fft = np.fft.ifftshift(fft_shift, axes=[0, 1])

filtered = cv2.idft(fft, flags=cv2.DFT_SCALE | cv2.DFT_REAL_OUTPUT)
mask_new = np.dstack((mask, np.zeros((image.shape[0], image.shape[1]), dtype=np.uint8)))

plt.figure()
plt.subplot(131)
plt.axis('off')
plt.title('original')
plt.imshow(image, cmap='gray')
plt.subplot(132)
plt.axis('off')
plt.title('no high frequencies')
plt.imshow(filtered, cmap='gray')
plt.subplot(133)
plt.axis('off')
plt.title('mask')
plt.imshow(mask_new*255, cmap='gray')
plt.tight_layout(True)
plt.show()
```

Image Thresholding

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

image = cv2.imread('../data/Lena.png', 0)

thr, mask = cv2.threshold(image, 200, 1, cv2.THRESH_BINARY)
print('Threshold used:', thr)

adapt_mask = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                   cv2.THRESH_BINARY_INV, 11, 10)


plt.figure(figsize=(10,4))
plt.subplot(131)
plt.axis('off')
plt.title('original')
plt.imshow(image, cmap='gray')
plt.subplot(132)
plt.axis('off')
plt.title('binary threshold')
plt.imshow(mask, cmap='gray')
plt.subplot(133)
plt.axis('off')
plt.title('adaptive threshold')
plt.imshow(adapt_mask, cmap='gray')
plt.tight_layout()
plt.show()
```

Binary operation

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

circle_image = np.zeros((500, 500), np.uint8)
cv2.circle(circle_image, (250, 250), 100, 255, -1)

rect_image = np.zeros((500, 500), np.uint8)
cv2.rectangle(rect_image, (100, 100), (400, 250), 255, -1)

circle_and_rect_image = circle_image & rect_image
circle_or_rect_image = circle_image | rect_image

plt.figure(figsize=(10,10))
plt.subplot(221)
plt.axis('off')
plt.title('circle')
plt.imshow(circle_image, cmap='gray')
plt.subplot(222)
plt.axis('off')
plt.title('rectangle')
plt.imshow(rect_image, cmap='gray')
plt.subplot(223)
plt.axis('off')
plt.title('circle & rectangle')
plt.imshow(circle_and_rect_image, cmap='gray')
plt.subplot(224)
plt.axis('off')
plt.title('circle | rectangle')
plt.imshow(circle_or_rect_image, cmap='gray')
plt.tight_layout(True)
plt.show()
```

Morphological Filter

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

image = cv2.imread('../data/Lena.png', 0)
_, binary = cv2.threshold(image, -1, 1, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

eroded = cv2.morphologyEx(binary, cv2.MORPH_ERODE, (3, 3), iterations=10)
dilated = cv2.morphologyEx(binary, cv2.MORPH_DILATE, (3, 3), iterations=10)

opened = cv2.morphologyEx(binary, cv2.MORPH_OPEN,cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)), iterations=5)
closed = cv2.morphologyEx(binary, cv2.MORPH_CLOSE,cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)),iterations=5)
grad = cv2.morphologyEx(binary, cv2.MORPH_GRADIENT,  cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5)))
```

```
plt.figure(figsize=(10,10))
plt.subplot(231)
plt.axis('off')
plt.title('binary')
plt.imshow(binary, cmap='gray')
plt.subplot(232)
plt.axis('off')
plt.title('erode 10 times')
plt.imshow(eroded, cmap='gray')
plt.subplot(233)
plt.axis('off')
plt.title('dilate 10 times')
plt.imshow(dilated, cmap='gray')
```

```
plt.subplot(234)
plt.axis('off')
plt.title('open 5 times')
plt.imshow(opened, cmap='gray')
plt.subplot(235)
plt.axis('off')
plt.title('close 5 times')
plt.imshow(closed, cmap='gray')
plt.subplot(236)
plt.axis('off')
plt.title('gradient')
plt.imshow(grad, cmap='gray')
plt.tight_layout()
plt.show()
```

다음의 프로그램을 작성하시오.

- Image Filtering
  - 영상을 화면에 출력하고 Unsharp mask를 적용한 결과를 출력하시오. (1)
  - (1)의 결과에 Sobel filter를 적용하고 출력하시오. (2)
  - (1)의 결과에 Gabor filter를 적용하고 출력하시오. (3)
  - (2)와 (3)의 차이를 Threshold의 변화에 따라서 출력하시오. (트랙바) (4)
  - (4)의 결과에 Opening과 Closing을 적용하시오.

- Frequency-based Filtering
  - 영상을 화면에 출력하고 DFT를 적용한 결과를 출력하시오.
  - 중심으로부터 원 모양과 사각형 모양의 필터링을 적용하고 결과를 출력하시오