数字图像处理及应用 第5次作业

组号： 16 小组成员： 冯坤龙 郝锦阳 朱从庆 辛梓阳 徐振良

Part I Exercises

Ex.1 Let A denote the set shown shaded in the following figure, and refer to the structuring elements shown (the black dots denote the origin). Sketch the result of the following operations:

(a) $(A \ominus B^4) \oplus B^2$.

(b) $(A \ominus B^1) \oplus B^3$.

(c) $(A \oplus B^1) \oplus B^3$.

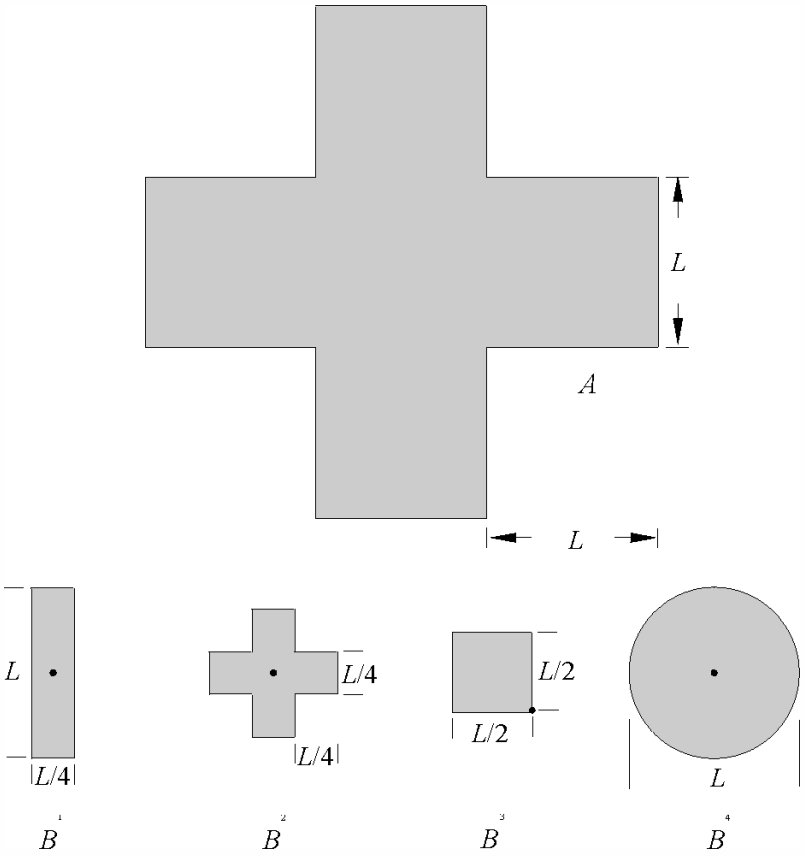
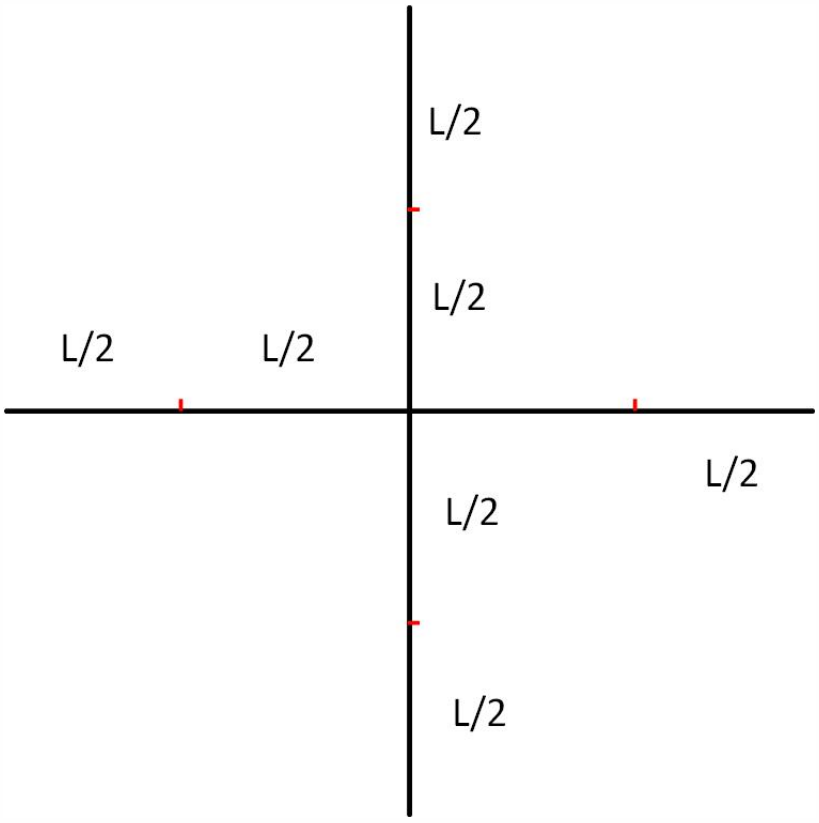
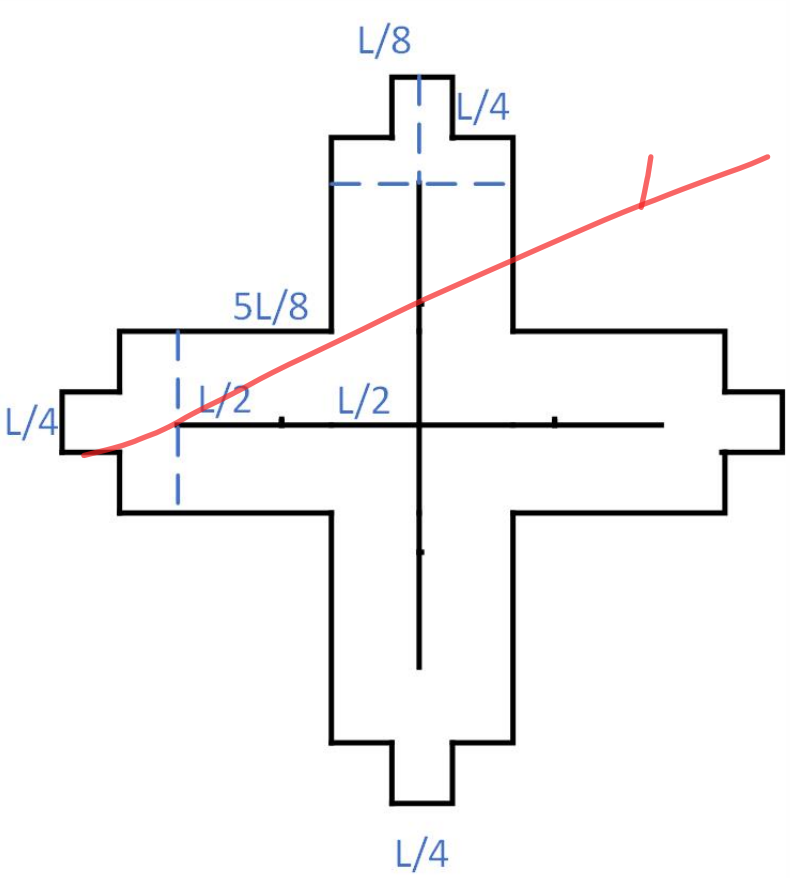


FIGURE 1 Image and structure elements

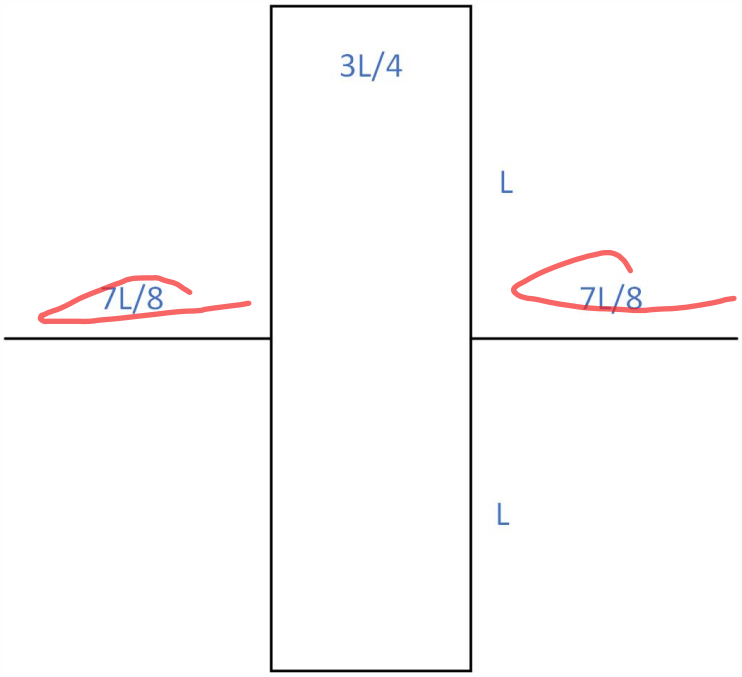
Answer:

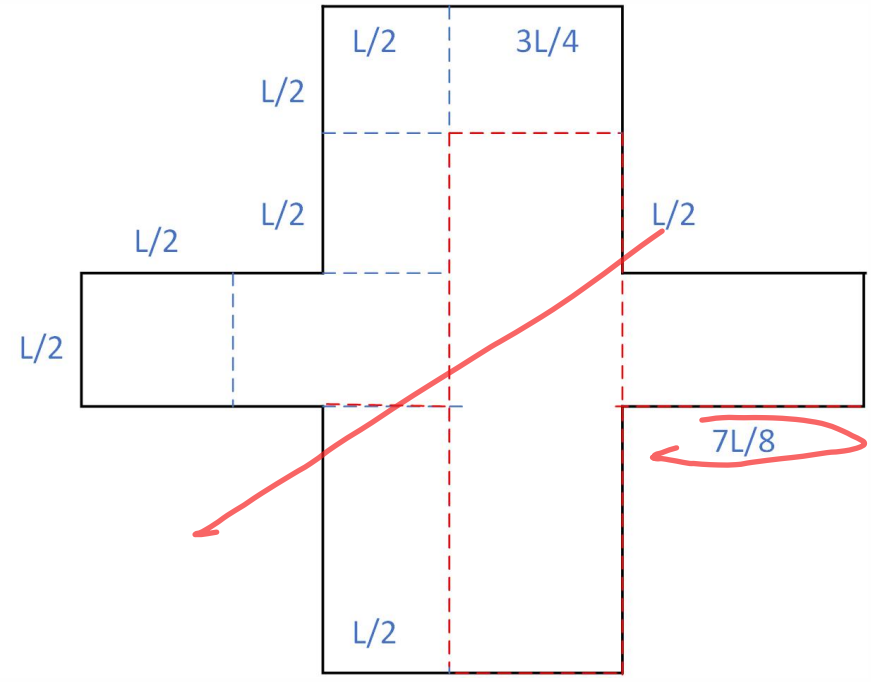
(a)



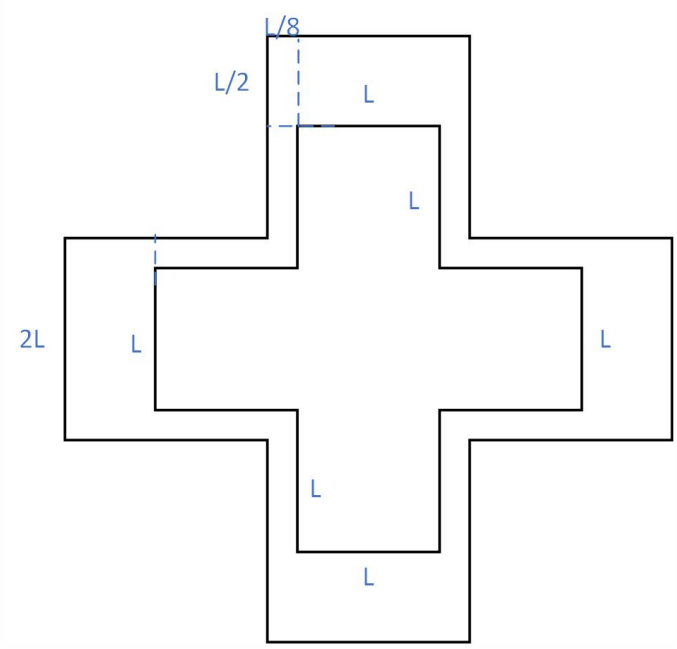


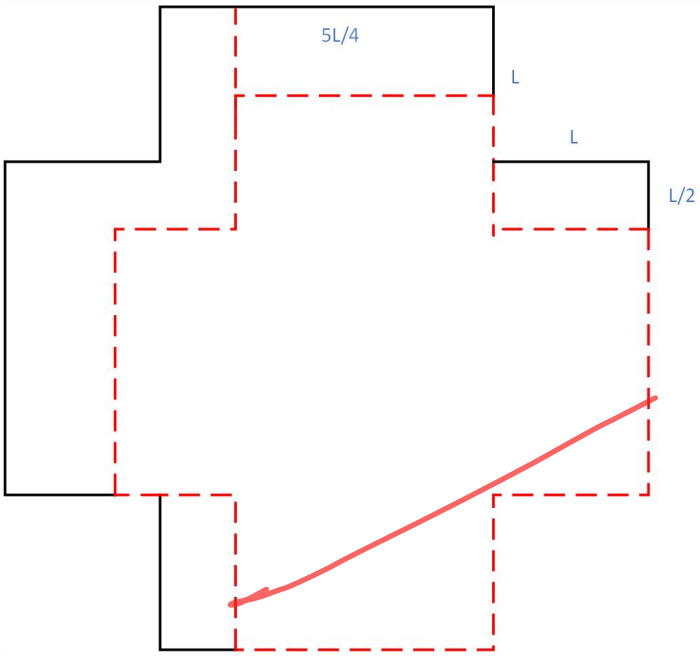
(b)





(c)





Ex.2 Prove the validity of the following expressions:
(**a**) $A \circ B$ is a subset (subimage) of A.
(**b**) If C is a subset of D, the $C \circ B$ is a subset of $D \circ B$.
(**c**) $(A \circ B) \circ B = A \circ B$.

Answer:

(a)Because

$$(B)_z \subseteq A, A \circ B = \cup\{(B)_z | (B)_z \subseteq A\}$$

We can conclude that A o B is a subset of A.

(b)From the definition of opening operation we know

$$C \circ B = \cup\{(B)_z | (B)_z \subseteq C\}$$
$$D \circ B = \cup\{(B)_z | (B)_z \subseteq D\}$$

If C is subset of D,the $C \circ B$ is a subset of $D \circ B$

(c)

$$(A \circ B) \circ B = [(A \ominus B \oplus B) \ominus B] \oplus B = [(A \ominus B) \cdot B] \oplus B$$

And

$$(A \ominus B) \bullet B \supseteq (A \ominus B)$$
$$(A \ominus B) \oplus B \supseteq (A \ominus B)$$
$$\Longrightarrow (A \circ B) \circ B \subseteq A \circ B$$

Because

$$A \circ B \subseteq (A \circ B) \circ B$$

We can get

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

Ex.3

(**a**) Give a morphological algorithm for converting an 8-connected binary boundary to an m-connected boundary. You may assume that the boundary is fully connected and that it is one pixel thick.
(**b**) Does the operation of your algorithm require more than one iteration with each structuring element? Explain your reasoning.
(**c**) Is the performance of your algorithm independent of the order in which the structuring elements are applied? If your answer is yes, prove it; otherwise give an example that illustrates the dependence of your procedure on the order of application of the structuring elements.

Answer:

(a) A is the input image containing the boundary

$$X_1 = A \otimes B_1$$
$$Y_1 = A \cap X_1^c$$
$$X_2 = Y_1 \otimes B_2$$
$$Y_2 = Y_1 \cap X_2^c$$
$$X_3 = Y_2 \otimes B_3$$
$$Y_3 = Y_2 \cap X_3^c$$
$$X_4 = Y_3 \otimes B_4$$
$$Y_4 = Y_3 \cap X_4^c$$

(b)Only one iteration is required. Application of the hit-or-miss transform using a given Bi finds all instances of occurrence of the pattern described by that structuring element.

0	.	x
.	.	0
x	0	0

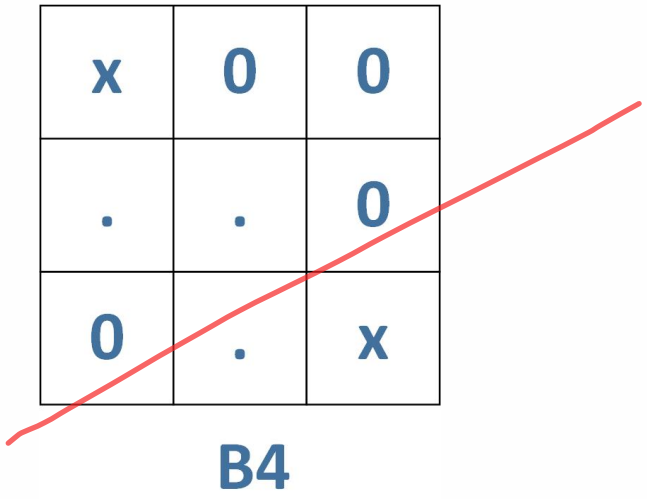
B1

x	.	0
0	.	.
0	0	x

B2

0	0	x
0	.	.
x	.	0

B3



（c）The order is important. The algorithm independent of the order in which the structuring elements are applied.

Ex.4 The rectangle in the binary image in FIGURE 2 is of size $m \times n$ pixels.
(**a**) What would the magnitude of the gradient of this image look like based on using the approximation given in Eq. (1)?

$$M(x, y) \approx |g_x| + |g_y| \tag{1}$$

Assume that g_x and g_y are obtained using the Sobel operators. Show all relevant different pixel values in the gradient image.
(**b**) Sketch the histogram of edge directions computed using Eq. (2). Be precise in labeling the height of each component of the histogram.

$$\alpha(x, y) = \arctan\left[\frac{g_y}{g_x}\right] \tag{2}$$

(**c**) What would the Laplacian of this image look like based on using the approximation in Eq. (3)? Show all relevant different pixel values in the Laplacian image.

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \tag{3}$$

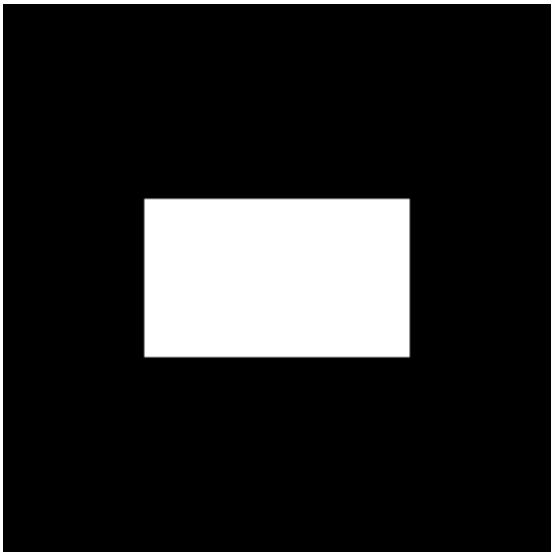


FIGURE 2 A binary image of size m × n

Answer:

(a)The intensity of this binary image is shown as below.

0	0	0	0	0	···	0	0	0	0	0
0	0	0	0	0	···	0	0	0	0	0
0	0	1	1	1	···	1	1	1	0	0
0	0	1	1	1	···	1	1	1	0	0
0	0	1	1	1	···	1	1	1	0	0
···										
0	0	1	1	1	···	1	1	1	0	0
0	0	1	1	1	···	1	1	1	0	0
0	0	1	1	1	···	1	1	1	0	0
0	0	1	1	1	···	1	1	1	0	0
0	0	0	0	0	···	0	0	0	0	0
0	0	0	0	0	···	0	0	0	0	0

Through the calculation of Sobel operator and Equ.1,the magnitude of this image can be obtained, shown as below.

g_y										
0	0	0	0	0	···	0	0	0	0	0
0	1	1	0	0	···	0	0	-1	-1	0
0	3	3	0	0	···	0	0	-3	-3	0
0	4	4	0	0	···	0	0	-4	-4	0
0	4	4	0	0	···	0	0	-4	-4	0
···										
0	4	4	0	0	···	0	0	-4	-4	0
0	4	4	0	0	···	0	0	-4	-4	0
0	4	4	0	0	···	0	0	-4	-4	0
0	3	3	0	0	···	0	0	-3	-3	0
0	1	1	0	0	···	0	0	-1	-1	0
0	0	0	0	0	···	0	0	0	0	0

g_x										
0	0	0	0	0	···	0	0	0	0	0
0	1	3	4	4	···	4	4	3	1	0
0	1	3	4	4	···	4	4	3	1	0
0	0	0	0	0	···	0	0	0	0	0
0	0	0	0	0	···	0	0	0	0	0
···										
0	0	0	0	0	···	0	0	0	0	0
0	0	0	0	0	···	0	0	0	0	0
0	0	0	0	0	···	0	0	0	0	0
0	-1	-3	-4	-4	···	-4	-4	-3	-1	0
0	-1	-3	-4	-4	···	-4	-4	-3	-1	0
0	0	0	0	0	···	0	0	0	0	0

$M(x,y)$										
0	0	0	0	0	···	0	0	0	0	0
0	2	4	4	4	···	4	4	4	2	0
0	4	6	4	4	···	4	4	6	4	0
0	4	4	0	0	···	0	0	4	4	0
0	4	4	0	0	···	0	0	4	4	0
···										
0	4	4	0	0	···	0	0	4	4	0
0	4	4	0	0	···	0	0	4	4	0
0	4	4	0	0	···	0	0	4	4	0
0	4	6	4	4	···	4	4	6	4	0
0	2	4	4	4	···	4	4	4	2	0
0	0	0	0	0	···	0	0	0	0	0

(b)From g_x,g_y and Equ.(2),after calculate we can get the angle value and quantities.We list them shown below.

angle	Quantity with this angle
0°	2(n-2)
18.4°	2
71.6°	2
90°	2(n-2)
-90°	2(n-2)
-71.6°	2
-18.4°	2

(c)The Laplacian's spatial mask and pixel values of this image by Laplacian are shown below.

0	1	0
1	-4	1
0	1	0

0	0	0	0	0	···	0	0	0	0	0
0	0	1	1	1	···	1	1	1	0	0
0	1	-2	-1	-1	···	-1	-1	-4	1	0
0	1	-1	0	0	···	0	0	-1	1	0
0	1	-1	0	0	···	0	0	-1	1	0
					···					
0	1	-1	0	0	···	0	0	-1	1	0
0	0	-1	0	0	···	0	0	-1	1	0
0	0	-1	0	0	···	0	0	-1	1	0
0	1	-2	-1	-1	···	-1	-1	-2	1	0
0	0	1	1	1	···	1	1	1	0	0
0	0	0	0	0	···	0	0	0	0	0

Ex.5 Marr and Hildreth noted that it is possible to approximate the LoG filter in Eq.(4) by a difference of Gaussians (DoG) in Eq.(5):

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4}$$

$$\mathrm{DoG}(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \tag{5}$$

with $\sigma_1 > \sigma_2$. To make meaningful comparisons between the LoG and DoG, the value of σ for the LoG must be selected as in the Eq.(6) so that the LoG and DoG have the same zero crossings,

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right] \tag{6}$$

(**a**) Derive Eq. (6).
(**b**) Let $k = \sigma_1/\sigma_2$ denote the standard deviation ratio discussed in connection with the DoG function, and express Eq. (6) in terms of k and σ_2.

Answer:

(a) Let Eq.(5)=0 $\Rightarrow$

$$\begin{aligned}
\frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} &= \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \\
\ln(\frac{1}{2\pi\sigma_1^2}) - \frac{x^2+y^2}{2\sigma_1^2} &= \ln(\frac{1}{2\pi\sigma_2^2}) - \frac{x^2+y^2}{2\sigma_2^2} \\
\ln(\frac{1}{2\pi\sigma_1^2}) - \ln(\frac{1}{2\pi\sigma_2^2}) &= \frac{x^2+y^2}{2\sigma_1^2} - \frac{x^2+y^2}{2\sigma_2^2} \\
\ln(\frac{1}{2\pi\sigma_1^2}) - \ln(\frac{1}{2\pi\sigma_2^2}) &= (x^2+y^2)(\frac{1}{2\sigma_1^2} - \frac{1}{2\sigma_2^2})
\end{aligned} \tag{5}$$

Let Eq.(4)=0 $\Rightarrow$

$$x^2 + y^2 = 2\sigma^2$$

From the above equation, it can be concluded that

$$\ln(\frac{1}{2\pi\sigma_1^2}) - \ln(\frac{1}{2\pi\sigma_2^2}) = \sigma^2(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2})$$

$$\sigma^2 = \frac{\ln(\frac{1}{2\pi\sigma_1^2}) - \ln(\frac{1}{2\pi\sigma_2^2})}{(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2})} = \frac{\ln(\frac{\sigma_2^2}{\sigma_1^2})}{\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}} = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_1^2} ln[\frac{\sigma_1^2}{\sigma_1^2}]$$

(b) let

$$\begin{aligned}
\sigma_1 &= k\sigma_2, (k > 1) \\
\sigma^2 &= \frac{k^2\sigma_2^4}{k^2\sigma_2^2 - \sigma_2^2} \ln[\frac{k^2\sigma_2^2}{\sigma_2^2}] \\
&= \frac{k^2}{k^2 - 1}\sigma_2^2 \ln(k^2)
\end{aligned}$$

Ex.6 An important area of application for image segmentation techniques is in processing images resulting from so-called *bubble chamber* events. These images arise from experiments in high-energy physics in which a beam of particles of known properties is directed onto a target of known nuclei. A typical event consists of incoming tracks, any one of which, in the event of a collision, branches out into secondary tracks of particles emanating

from the point of collision. Propose a segmentation approach for detecting all tracks angled at any of the following six directions off the horizontal: $\pm 25^\circ$, $\pm 50^\circ$, and $\pm 75^\circ$. The allowed estimation error in any of these six directions is $\pm 5^\circ$. For a track to be valid it must be at least 100 pixels long and have no more than three gaps, each not exceeding 10 pixels. You may assume that the images have been preprocessed so that they are binary and that all tracks are 1 pixel thick, except at the point of collision from which they emanate. Your procedure should be able to differentiate between tracks that have the same direction but different origins. (**Hint: Base your solution on the Hough transform.**)

Answer:

First, the axis is divided into six subregions corresponding to the Angle between the orbit and the horizontal direction in six directions. Since the maximum is PI, the range is PI.
Secondly, the axis ranges in

$$[-\sqrt{2D}, \sqrt{2D}]$$

The axis requires subtle partitioning to distinguish trajectories with the same direction but different starting points.
Finally, the Hough transform can be used as a "filter" to divide all the points in the image into six groups, categorized according to six directions. Then test whether the trajectories meet the requirements of the problem according to different groups.

Part II Programming

1. Refer to the image and the disk structuring element shown in FIGURE 3. Sketch what the sets C, D, E, and F would look like for the following sequence of operations:

a) $C = A \ominus B$;

b) $D = C \oplus B$;

c) $E = D \oplus B$;

d) $F = E \ominus B$.

Set A consists of all the foreground pixels (white), except the structuring element, B, which you may assume is just large enough to encompass any of the random elements in the image. Note that the sequence of operations above is simply the opening of A by B followed by a closing of the result by B.

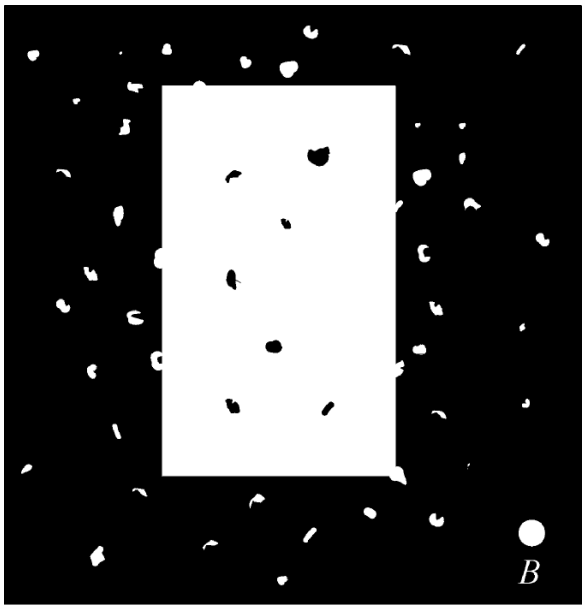
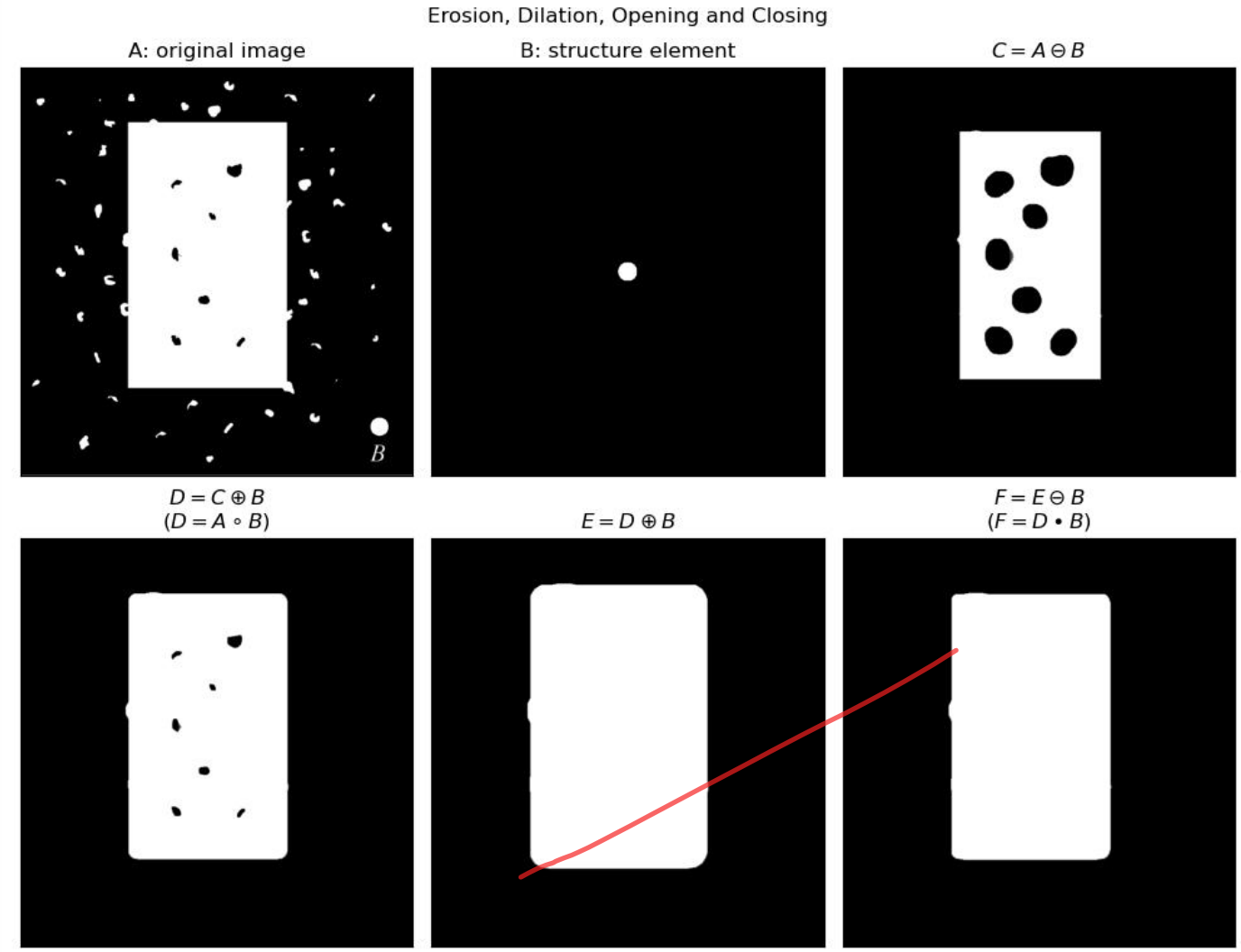


FIGURE 3 Image and the disk structuring element

(*followed by* ***Matlab live Scripts*** *or* ***Jupyter Scripts*** *and running results*)

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

original_img = cv2.imread('../images/Figp0917.png', 0)

# create a circle structure element with radius 16
se = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (32, 32))

A = original_img
B = se.copy()
# erode A by B using function `cv2.erode()`
C = cv2.erode(A, B)
# dilate C by B using function `cv2.dilate()`
```

```
D = cv2.dilate(C, B)
# dilate D by B
E = cv2.dilate(D, B)
# erode E by B
F = cv2.erode(E, B)

# display the results
B = np.zeros(original_img.shape)
h, w = B.shape[:2]
sh, sw = se.shape[:2]
y = int((h - sh) / 2)
x = int((w - sw) / 2)
B[y:y + sh, x:x + sw] = se

disp_img = [A, B, C, D, E, F]
disp_cap = ["A: original image", "B: structure element", "$C=A\ominus B$",
"$D=C\oplus B$\n$(D=A\circ B)$",
            "$E=D\oplus B$", "$F=E\ominus B$\n" + r"$(F=D\bullet B)$"]

fig, axs = plt.subplots(2, 3, figsize=(10, 8))
for i in range(len(disp_img)):
    ax = axs.flat[i]
    ax.imshow(disp_img[i], 'gray')
    ax.set_title(disp_cap[i])
    ax.set_xticks([])
    ax.set_yticks([])
plt.suptitle("Erosion, Dilation, Opening and Closing")
plt.tight_layout()

# save
# output = f'../images/Erosion, Dilation, Opening and Closing.jpg'
# plt.savefig(output)

plt.show()

```



2. Consider the image in FIGURE 4, which shows a region of small circles enclosed by a region of larger circles. (**a**) Give a morphologic algorithm to partition the image into two parts, in which one contains small circles and another contains larger circles. You can make any assumptions that you need to make for the method to work. (**b**) Sketch the result in each step of your algorithm.

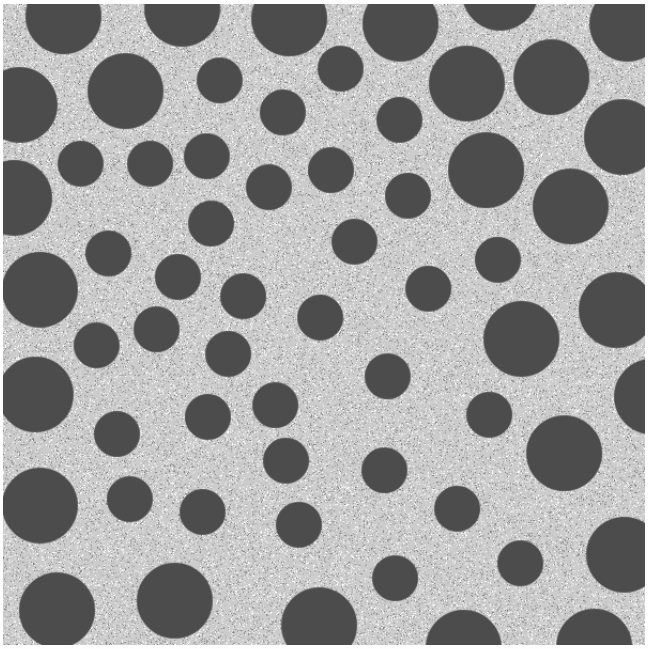
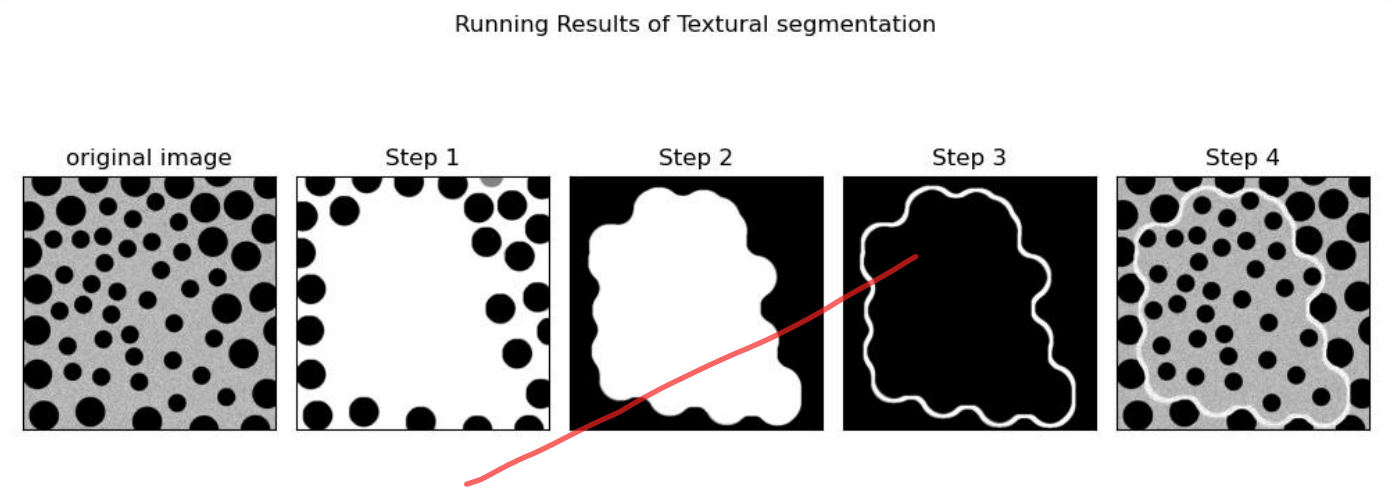


FIGURE 4

(*followed by* ***Matlab live Scripts*** *or* ***Jupyter Scripts*** *and running results*)

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

# open
original_image = cv2.imread("../images/FigP0934.png", flags=0)
target_image = []

middle_size = (50, 50)
middle_element = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, middle_size)
target_image.append(cv2.morphologyEx(original_image, cv2.MORPH_CLOSE,
middle_element))

large_size = (100, 100)
large_element = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, large_size)
target_image.append(cv2.morphologyEx(target_image[0], cv2.MORPH_OPEN,
large_element))

cut_size = (10, 10)
element = cv2.getStructuringElement(cv2.MORPH_RECT, cut_size)
target_image.append(cv2.morphologyEx(target_image[1], cv2.MORPH_GRADIENT,
element))

target_image.append(cv2.bitwise_or(original_image, target_image[2]))

# display the results
fig, axs = plt.subplots(nrows=1, ncols=5, figsize=(10, 4))
ax = axs[0]
ax.imshow(original_image, cmap='gray'), ax.set_title(f"original image"),
ax.set_xticks([]), ax.set_yticks([])
for i in range(4):
    ax = axs[i + 1]
    ax.imshow(target_image[i], 'gray')
    ax.set_title(f"Step {i + 1}")
    ax.set_xticks([]), ax.set_yticks([])

plt.suptitle("Running Results of Textural segmentation")
plt.tight_layout()

# save
# output = f'../images/Textural segmentation.jpg'
# plt.savefig(output)

plt.show()

```



3. The objects and background in FIGURE 5 have a mean intensity of 170 and 60, respectively, on a [0, 255] scale. The image is corrupted by Gaussian noise with 0 mean and a standard deviation of 10 intensity levels.

(**a**) Segment the image based on thresholding (Refer to Example 10.15 in textbook, and pay attention to the choice of initial threshold T).

(**b**) Repeat segmentation based on region growing. (**Optional**)

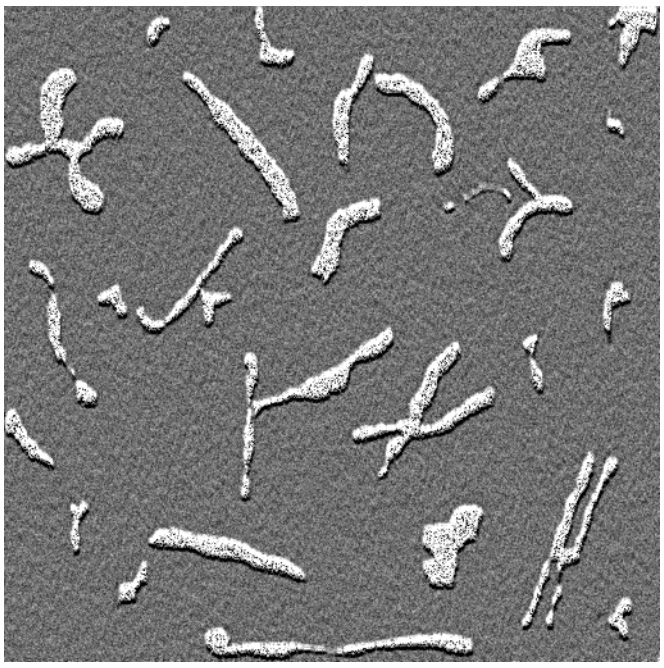


FIGURE 5

(*followed by* ***Matlab live Scripts*** *or* ***Jupyter Scripts*** *and running results*)

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

original_img = cv2.imread("../images/FigP1036.png", 0)
thresholded_img = cv2.threshold(original_img, 60, 255, cv2.THRESH_OTSU)[1]

plt.figure(figsize=(8, 6))
plt.subplot(221), plt.imshow(original_img, 'gray'), plt.title("original")

```

```
plt.subplot(222), plt.imshow(thresholded_img, 'gray'), plt.title("thresholded")

plt.subplot(212), plt.hist(original_img.flatten(), 256, [0, 255], log=True)
plt.tight_layout()

# save
# output = f'../images/Thresholding segment.jpg'
# plt.savefig(output)

plt.show()

```