(CSE 4734: Digital Image Processing Lab)

LAB 4 Report

Submitted by:
Md. Mosharrifur Rahman Ratul
Student ID: 210041213
Section: 2A
BSc in CSE, Dept of Computer Science and Engineering
Islamic University of Technology (IUT)

January 4, 2026

Contents

1 Task 1

1.1 Problem Statement

You are required to implement two functions that will simulate the behaviour of morphological Dilation and Erosion. For each of these functions, we will be considering a parameter, SE, which denotes the structuring element which we will need to preconstruct before passing them to the corresponding function.

1.2 Approach

In this task, morphological erosion and dilation are implemented using a predefined structuring element (SE). Both circular and rectangular structuring elements are constructed and then slid over the input image. For erosion, the minimum pixel value within the region defined by the SE is selected, whereas for dilation, the maximum value is chosen. Padding is applied to the image boundaries to ensure that the filtering operation can be performed at the edges without reducing the image size.

1.3 Code Implementation

The python code of the functions for erosion , dilation , rectangular and circular Structural element with various size is given below :-

```
import numpy as np

def dilation(image, SE):

    img_h, img_w = image.shape
    se_h, se_w = SE.shape

    pad_h = se_h // 2
    pad_w = se_w // 2

```

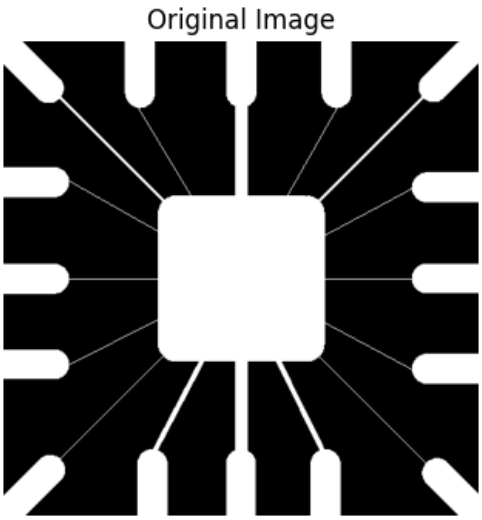
```
    padded = np.pad(image, ((pad_h, pad_h), (pad_w, pad_w)),
        mode='reflect')

    dilated_image = np.zeros_like(image)

    for i in range(img_h):
        for j in range(img_w):
            region = padded[i:i+se_h, j:j+se_w]
            dilated_image[i, j] = np.max(region[SE == 1])

    return dilated_image

def erosion(image, SE):

    img_h, img_w = image.shape
    se_h, se_w = SE.shape

    pad_h = se_h // 2
    pad_w = se_w // 2

    padded = np.pad(image, ((pad_h, pad_h), (pad_w, pad_w)),
        mode='reflect')

    eroded_image = np.zeros_like(image)

    for i in range(img_h):
        for j in range(img_w):
            region = padded[i:i+se_h, j:j+se_w]
            eroded_image[i, j] = np.min(region[SE == 1])

    return eroded_image

def circular_SE(radius):
    y, x = np.ogrid[-radius:radius+1, -radius:radius+1]
    SE = ((x**2 + y**2) <= radius*radius).astype(np.uint8)
    return SE

def rectangular_SE(size):
    SE = np.ones((size, size), dtype=np.uint8)
    return SE
```

For the output of dilation and erosion , I used following code part with various SE for analysis:-

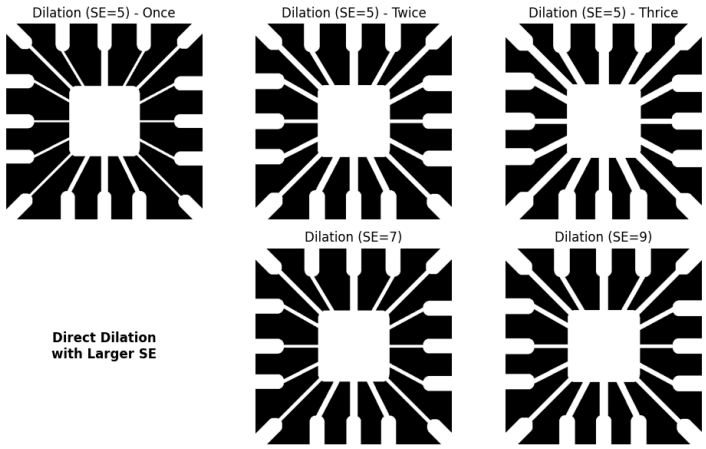
```

from PIL import Image, ImageFilter
import numpy as np
import matplotlib.pyplot as plt

img = Image.open("1.tif").convert("L")
```

```
gray_img = np.array(img)

SE = rectangular_SE(5)

dilated_1 = dilation(gray_img, SE)
dilated_2 = dilation(dilated_1, SE)
dilated_3 = dilation(dilated_2, SE)

SE_7 = rectangular_SE(7)
SE_9 = rectangular_SE(9)

dilated_4 = dilation(gray_img, SE_7)
dilated_5 = dilation(gray_img, SE_9)

gray_img2 = np.array(img)
eroded_1 = erosion(gray_img2, SE)
eroded_2 = erosion(eroded_1, SE)
eroded_3 = erosion(eroded_2, SE)

eroded_4 = erosion(gray_img2, SE_7)
eroded_5 = erosion(gray_img2, SE_19)
```

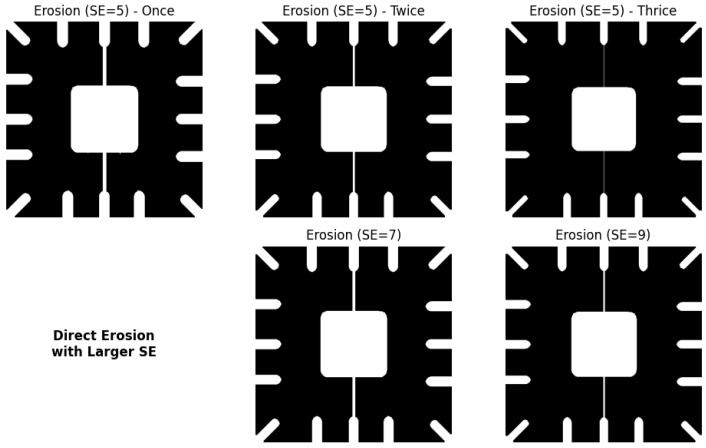
1.4 Output Result



Original Input Image



Dilation Results Using Repeated and Larger Structuring Elements



Erosion Results Using Repeated and Larger Structuring Elements

1.5 Result Analysis

Morphological dilation and erosion modify the shape and structure of objects in an image based on a given structuring element (SE). The experimental results clearly

demonstrate how repeated applications of these operations relate to the size of the structuring element.

When dilation is applied multiple times using the same structuring element, the bright regions in the image continue to expand. Each successive dilation increases object boundaries outward by an amount determined by the SE shape, it is usually the half of the SE size. As observed, applying dilation twice with a 5×5 structuring element produces a result that is visually similar to applying dilation once with a larger structuring element (such as 7×7). While the consecutive 2 dilation with SE size 5 is equivalent to the Dilation with large structural element of size 9×9

Similarly, when erosion is applied multiple times using the same structuring element, the bright regions gradually shrink while dark regions expand. Applying erosion twice with a 5×5 structuring element removes thin structures and reduces object size more aggressively. And the consecutive erosion with SE size 5 for twice and thrice is equivalent to erosion with large SE size 7 and 9 respectively.

Overall, these results confirm that multiple applications of dilation or erosion with a fixed structuring element are equivalent to a single operation using a larger structuring element. This property is fundamental in morphological image processing and is useful for controlling the degree of shape expansion or contraction in an image.

2 Task 2

2.1 Problem Statement

Granulometry is a field that deals with determining the size distribution of particles in an image. Determine the sizes of granules in Fig0941(a)(wood_dowels).tif which is an image of wood dowel plugs of two dominant sizes.

Your task is to write a function which should perform opening with circular SEs of variable sizes and generate a graph plot showing the difference in surface area vs radius of SE. It is recommended to apply smoothing on the given image before you apply the Opening operations.

2.2 Approach

In this task, granulometry is performed to analyze the size distribution of objects present in the image using morphological operations. Initially, the grayscale image is smoothed using a Gaussian filter to reduce noise and small intensity variations. The smoothed image is then converted into a binary image using a fixed threshold so that the objects are clearly separated from the background.

Morphological opening is applied repeatedly using circular structuring elements of increasing radius from 1 to 30. Since opening removes objects smaller than the structuring element, this process gradually eliminates smaller structures from the image. For each structuring element size, the total intensity of the opened image is calculated and stored. Additionally, the difference between consecutive intensity values is computed to observe how much image content is removed as the

structuring element size increases. These measurements collectively help in understanding the dominant object sizes present in the image. At the point where is a major change in the intensity differences , it indicates a group of objects is removed and the current Structural Element size is the size of the object.

2.3 Code Implementation

The functions for opening, closing and granulometry calculation are following:-

```
1
2  def opening(image, SE):
3      eroded = erosion(image, SE)
4      opened = dilation(eroded, SE)
5      return opened
6
7  def closing(image, SE):
8      dilated = dilation(image, SE)
9      closed = erosion(dilated, SE)
10     return closed
11
12 def granulometry(image):
13
14     smooth = Image.fromarray(image).filter(ImageFilter.
           GaussianBlur(radius=1))
15     smooth = np.array(smooth)
16
17     binary_img = (smooth > 128).astype(np.uint8)
18
19     intensities = []
20     differences = []
21     se_sizes = list(range(1, 31))
22
23     prev_intensity = None
24
25     for r in se_sizes:
26         SE = circular_SE(r)
27         opened = opening(binary_img, SE)
28
29         total_intensity = np.sum(opened)
30         intensities.append(total_intensity)
31
32         if prev_intensity is not None:
```

```
            differences.append(prev_intensity - total_intensity
                )

        prev_intensity = total_intensity

    return intensities, differences, se_sizes, smooth,
        binary_img
```

2.4 Output Result

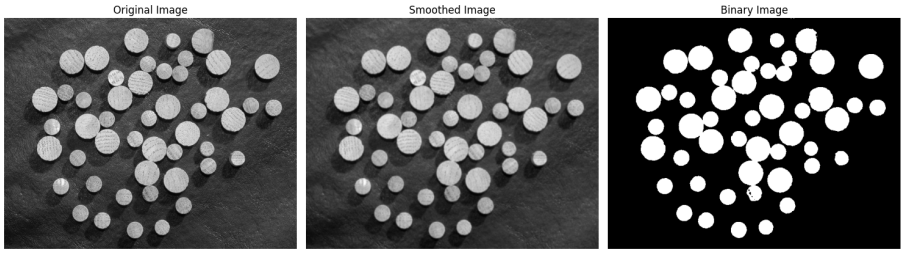


Figure 1: Original image, smoothed image, and binary image used for granulometry

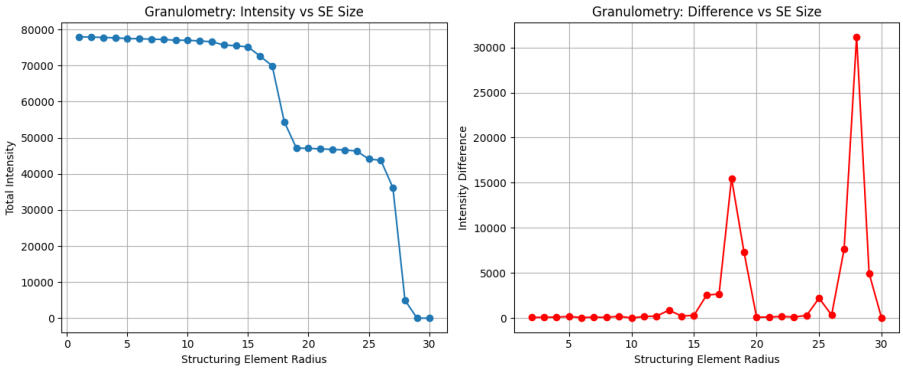


Figure 2: Granulometry plots: total intensity and intensity difference versus structuring element size

2.5 Result Analysis

The results show that the total image intensity decreases as the size of the structuring element increases. This occurs because morphological opening removes objects that

are smaller than the current structuring element. When smaller structuring elements are used, most objects remain in the image, resulting in higher intensity values. As the structuring element radius increases, smaller objects are progressively removed, causing a gradual reduction in total intensity.

Sometimes the total intensity curve decreases slowly and some point there is a sharp decrease. The reason for this is that , the very slow change actually removes the small noise or particles from the image. While the large drop of intensity indicates that a group of obect is removed. Thus the size of the objects are determined from the SE size.

The intensity difference plot provides clearer insight into the object size distribution. Peaks in the difference curve indicate structuring element sizes at which a significant amount of image object is removed. These peaks correspond to dominant object sizes in the image. Once the structuring element becomes larger than the largest objects present, the intensity changes become minimal, indicating that most objects have already been removed. Here, from the previous we found that the dominant object sizes are of 18 unit and 28 unit radius.

Overall, granulometry effectively reveals the size characteristics of objects in the image. The method demonstrates how morphological opening with varying structuring element sizes can be used as a simple yet powerful tool for size-based image analysis.

3 Task 3

3.1 Problem statement

Consider the image given in "FigP0934 (blobs_in_circular _arrangement) .tif", which shows a region of small circles enclosed by a region of larger circles. Locate the boundary between those distinct texture regions.

3.2 Approach

The input image is first converted to a binary image to separate the circular patterns from the background with black and white, here all darker shade is converted to complete black and brighter shade into white.
Morphological opening with circular structuring elements of increasing radius is applied to remove the smaller inner circles. So, I repeatedly performed opening by increasing the circular SE size and found the desired radius of inner circle.
Once the radius is sufficient to eliminate the small textures, morphological closing with larger structuring elements is used to highlight the outer circular region. Then similarly by closing I detect the outer boundary and with the required radius I have found the boundary of larger circles.

3.3 Code Implementation

At first to find the required inner circles radius following code is used :-

```
1
2 img = Image.open("3.tif").convert("L")
3 gray_img = np.array(img)
4
5 binary_img = (gray_img < 128).astype(np.uint8)
6
```

```
radii = list(range(14, 26, 2))

plt.figure(figsize=(12, 8))

for idx, r in enumerate(radii):
    SE = circular_SE(r)
    opened_img = opening(binary_img, SE)

    plt.subplot(2, 3, idx + 1)
    plt.imshow(opened_img, cmap='gray')
    plt.title(f"Opening (r = {r})")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Then to find the required radius for properly identifying the outer boundary of larger circles the following code part was used:-

```

SE = circular_SE(23)
opened_img = opening(binary_img, SE)

radii_outer = list(range(32, 44, 2))

plt.figure(figsize=(12, 8))

for idx, r in enumerate(radii_outer):
    SE = circular_SE(r)

    dilated = dilation(opened_img, SE)
    closed = erosion(dilated, SE)

    plt.subplot(2, 3, idx + 1)
    plt.imshow(closed, cmap='gray')
    plt.title(f"Closing (r = {r})")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Finally to detect the boundary the following function is used :-

```

def locateOuterBoundary(inner_radius,outer_radius):
  required_radius = outer_radius

```

```
    img = Image.open("3.tif").convert("L")
    gray_img = np.array(img)
    binary_img = (gray_img < 128).astype(np.uint8)

    SE = circular_SE(inner_radius)
    opened_img = opening(binary_img, SE)

    SE = circular_SE(required_radius)

    closed = closing(opened_img, SE)

    SE = circular_SE(5)

    final_close = erosion(closed, SE)
    boundary = final_close - closed

    return boundary
```

3.4 Output Result

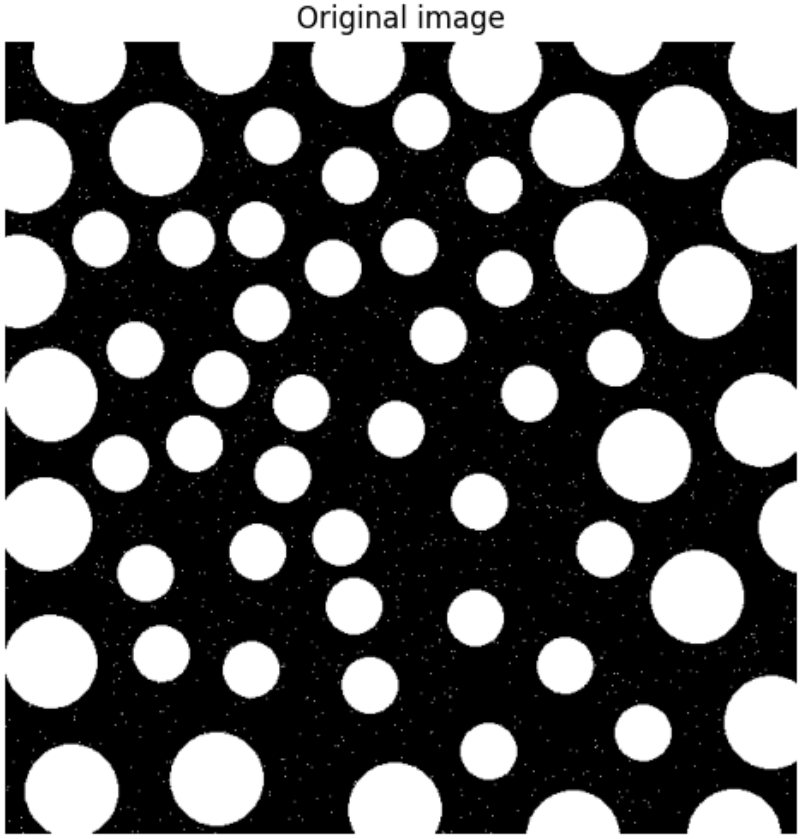


Figure 3: Original input image

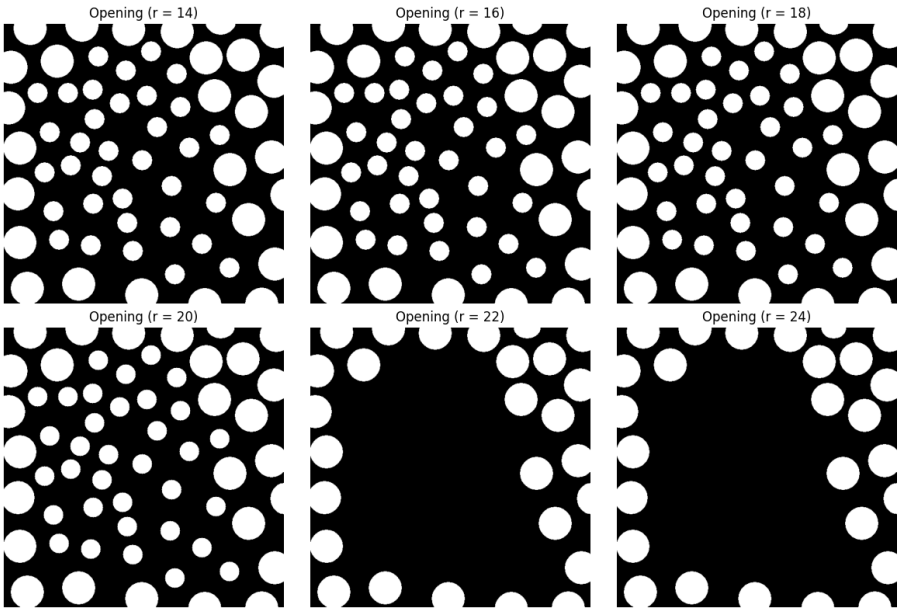


Figure 4: Opening result with various SE size

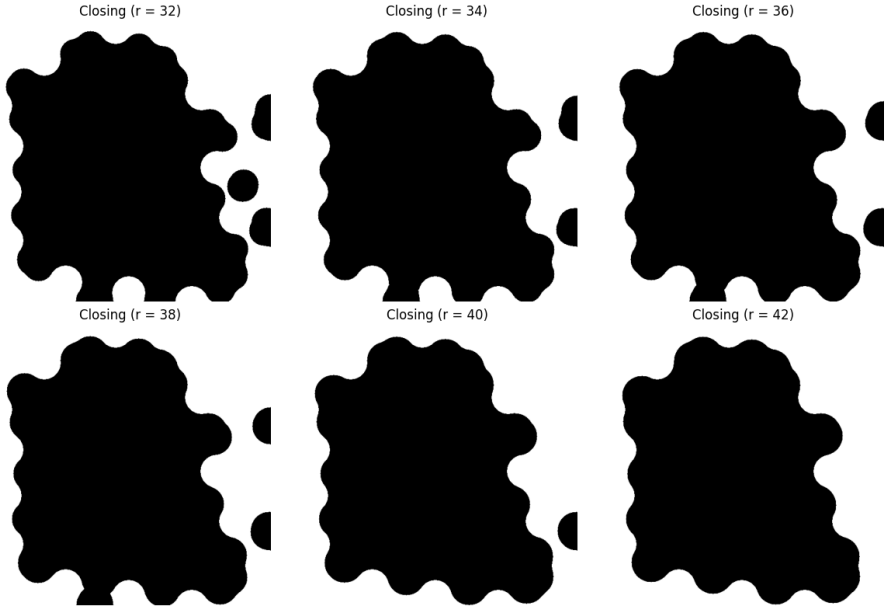


Figure 5: Closing result for larger circles boundary area with various SE size

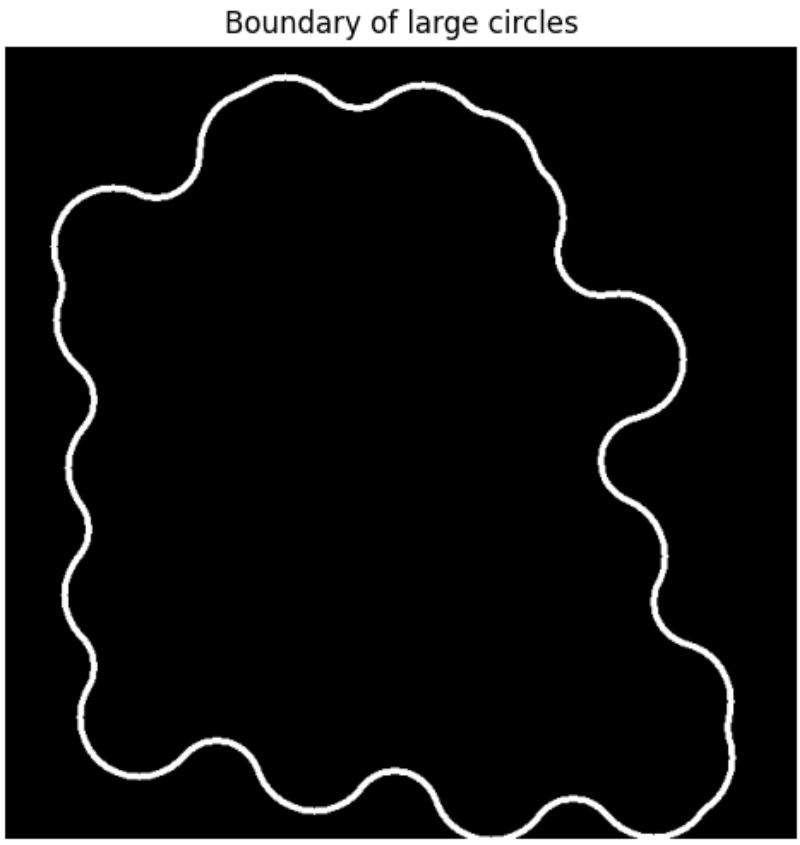


Figure 6: The required output boundary of larger circles

3.5 Result Analysis

The objective of this task was to locate the boundary between two distinct texture regions consisting of small and large circular patterns. Morphological operations were employed to exploit the difference in object size between these two regions.

From the opening results using circular structuring elements of increasing radius, it is observed that small circular objects gradually disappear as the radius of the structuring element increases. When the structuring ele-

ment radius becomes larger than the radius of the inner small circles, these objects are completely removed while the larger outer circles remain mostly unaffected. This confirms that morphological opening effectively removes structures that are smaller than the chosen structuring element. We found that the radius of inner small circles are 21 as opening with SE size 22 removes them.

After identifying a suitable radius for eliminating the inner region, morphological closing was applied using larger structuring elements. Closing fills gaps and connects nearby bright regions, which helps in consolidating the outer circular texture into a single continuous region. As the structuring element radius increases, the outer region becomes smoother and more uniform, making the boundary between the two texture regions more distinguishable. Here, the required SE size for closing to properly find the boundary is taken as 42.

Finally, boundary detection was performed by applying an additional erosion followed by subtracting the closed image from the eroded result. This subtraction highlights the pixels corresponding to the transition between the inner and outer regions, effectively extracting the boundary. The resulting boundary accurately separates the small-circle region from the large-circle region.

Overall, the results demonstrate that morphological opening and closing, when combined with appropriately chosen circular structuring elements, provide an effective method for texture-based boundary detection. The success of the approach relies on selecting structuring element sizes that correspond to the scale of the texture patterns present in the image.

4 Task 4

4.1 Problem Statement

A preprocessing step in an application of microscopy is concerned with the issue of isolating individual round particles from similar particles that overlap in groups of two or more particles as shown in the image FigP0936 (bubbles_on_black_background).tif.

Assuming that all particles are of the same size, propose a morphological algorithm that produces three images consisting respectively of

- Only of particles that have merged with the boundary of the image.
- Only overlapping particles.
- Only nonoverlapping particles.

4.2 Approach

First, the image was converted to binary so that particles are clearly distinguished from the background. The circles radius also have found by consecutive erosion.
Particles touching the boundary were identified using morphological dilation starting from the image edges.
After removing the boundary-touching particles, the remaining connected components were labeled. By comparing the size of each component to a chosen threshold which is the area of a circle from the found radius, larger components were classified as overlapping particles and smaller ones as non-overlapping single particles. This approach relies on the fact that overlapping particles form larger connected regions than individual particles of the same size, allowing a clear separation.

4.3 Code Implementation

The code part for finding the circle radius with erosion :-

```
img = Image.open("4.tif").convert("L")
gray_img = np.array(img)
binary_img = (gray_img > 128).astype(np.uint8)

radii = list(range(6, 15, 1))

plt.figure(figsize=(12, 18))

for idx, r in enumerate(radii):
    SE = circular_SE(r)
    opened_img = erosion(binary_img, SE)

    plt.subplot(3, 3, idx + 1)
    plt.imshow(opened_img, cmap='gray')
    plt.title(f"Erosion (radius = {r})")
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Then for finding the three required image following code portion is used:-

```
from scipy.ndimage import label

def particles_touching_boundary(binary_img):
    h, w = binary_img.shape

    boundary = np.zeros_like(binary_img)
    boundary[0, :] = 1
    boundary[-1, :] = 1
    boundary[:, 0] = 1
    boundary[:, -1] = 1

    touching = binary_img * boundary

    SE = circular_SE(3)

    prev = np.zeros_like(binary_img)
    curr = touching.copy()

    while not np.array_equal(prev, curr):
```

```
        prev = curr.copy()
        curr = dilation(curr, SE) * binary_img

    return curr

boundary_particles = particles_touching_boundary(binary_img)


img = Image.open("4.tif").convert("L")
binary = np.array(img)
binary = (binary > 128).astype(np.uint8)
binary_img = binary - boundary_particles

labeled, num = label(binary_img)

overlapping = np.zeros_like(binary_img)
non_overlapping = np.zeros_like(binary_img)

AREA_THRESHOLD = 530

for lab in range(1, num + 1):
    component = (labeled == lab)
    area = np.sum(component)

    if area > AREA_THRESHOLD:
        overlapping[component] = 1
    else:
        non_overlapping[component] = 1
```

4.4 Output Result

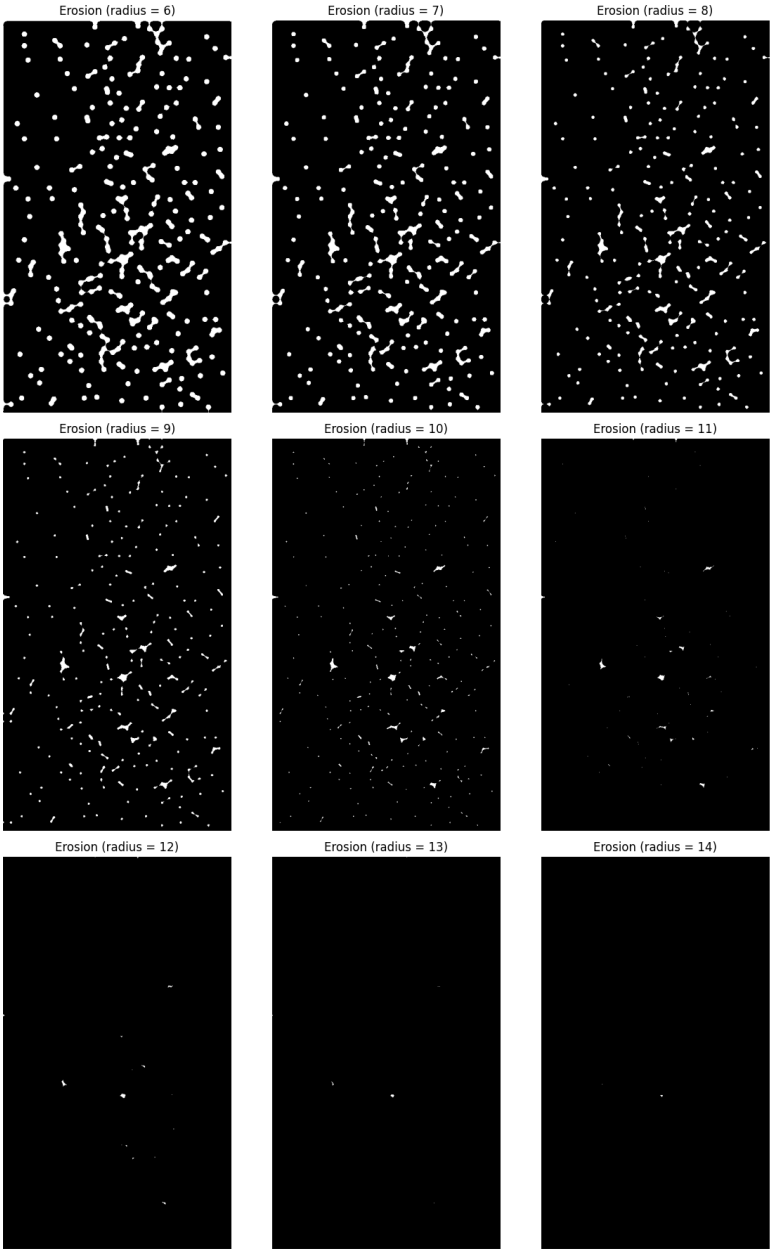


Figure 7: Erosion result with various size SE

Thus , we get the size of the circle's radius is about 13 unit, as SE size 14 removes them.

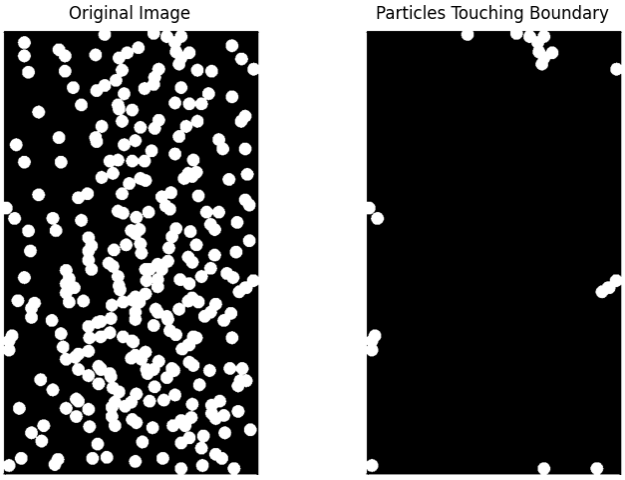


Figure 8: Original Image and Boundary particles

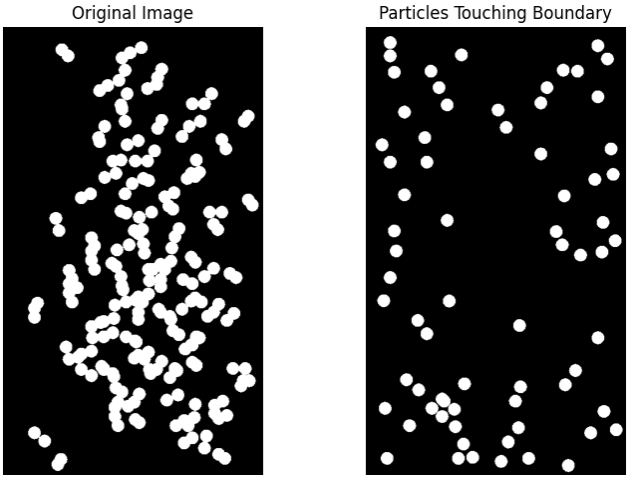


Figure 9: Overlapping and Non-overlapping particles

4.5 Result Analysis

The results show that the morphological and connected-component approach effectively separates the three categories of particles. The boundary-touching particles were successfully detected by propagating the edges using dilation, ensuring that all particles connected to the image boundary were included. Here, as we start dilation from the image boundary only the pixel with bright value will be dilated or which are actually the circles. Thus , we can separate the boundary touching particles.
Overlapping particles were correctly identified by analyzing the area of each connected component; because overlapping particles form larger connected regions, any component exceeding a threshold could be reliably classified as overlapping. Here, by successive erosion of the image, we found that all the circles have radius equal to 13, as the circles removed while we perform erosion with structural element of 14×14. Thus, we find the area of a single circle using the formula :-

$$\begin{aligned} Area &= \pi r^2 \\ &= \pi \times 13^2 \\ &= 169\pi \\ &\approx 530.93 \end{aligned}$$

Thus, if the connected component size is greater than the area of a single circle, it must have more than one circle or overlapped circles. Otherwise , they could be fall into non-overlapped particles. Thus overlapped and non-overlapped particles are founded.
The method works well because all particles are approximately the same size, which makes area-based classification reliable. Overall, the approach is simple, robust,

and does not depend on the exact shape of the particles, making it suitable for preprocessing in microscopy images.