Assignment 9

- Develop and apply erosion and dilation to two images
- Apply opening to an image
- Apply closing to an image

Erosion

Erosion is an operation about reducing edges in the image by using the kernel. The kernel size affects the area which will be deleted (The white area is reduced). Therefore, it is used for removing noise. Moreover, it can separate the bridge of objects if the kernel is big enough. In the erosion, we need to match the kernel with the binary image. After overlaying the kernel onto the image, if the pixels are a perfect match (fit) the eroded image is considered as 1. Otherwise, they are considered as 0.

Dilation

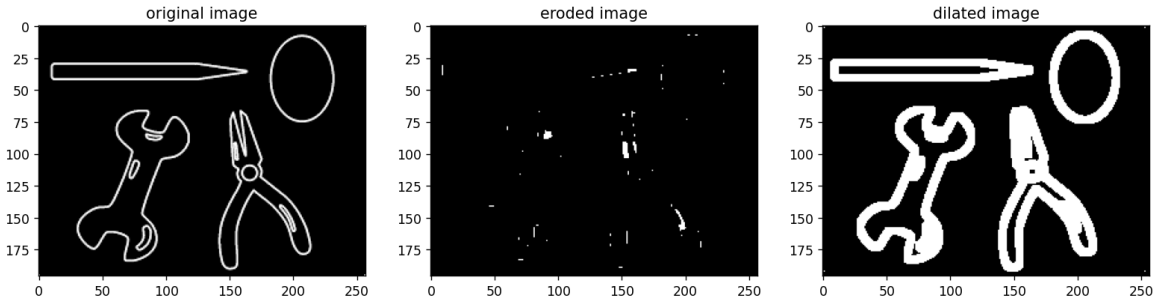
Dilation is the opposite operation of erosion. Dilation expands the edge of the image (The white area is increased). Therefore, it is used for filling the holes and gaps in the image. Moreover, it can make the edge of the image thicker. In dilation, we also match the kernel with the image as well. In this time, the pixels which are not matched will be considered as 0 and the rest are 1.

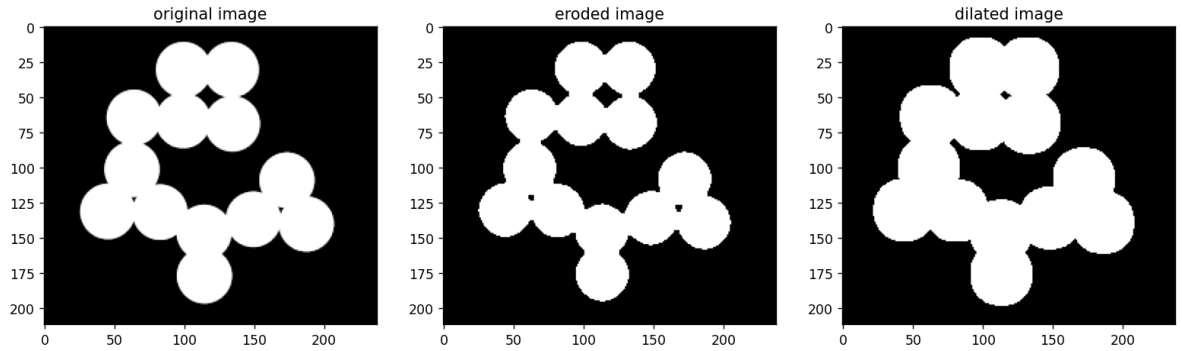
Coding:

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

img = cv2.imread('tools.png', 0)
(thresh, img2) = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY)
img2 = img2/255     # normalize to binary 0,1 image

h,w = img.shape

# Erosion
def erosion(img,h,w,kernel):
    s_element = np.ones((kernel,kernel))
    img_erosion= np.zeros((h,w))
    x  = int((kernel-1)/2)

    for i in range(x, h-x):
        for j in range(x, w-x):
            temp = img[i-x:i+x+1, j-x:j+x+1]
            product = temp*s_element
            img_erosion[i-1,j-1]= np.min(product)
    return img_erosion

```

```python
# Dilation
def dilation(img,h,w,kernel):
    s_element = np.ones((kernel,kernel))
    img_dilation= np.zeros((h,w))
    x  = int((kernel-1)/2)

    for i in range(x, h-x):
        for j in range(x, w-x):
            temp = img[i-x:i+x+1, j-x:j+x+1]
            product = temp*s_element
            img_dilation[i-1,j-1]= np.max(product)
    return img_dilation

eroded_img = erosion(img2,h,w,3)
dilated_img = dilation(img2,h,w,5)

plt.subplot(131)
plt.title('original image')
plt.imshow(img,cmap="gray")

plt.subplot(132)
plt.title('eroded image')
plt.imshow(eroded_img,cmap='gray')

plt.subplot(133)
plt.title('dilated image')
plt.imshow(dilated_img,cmap='gray')

plt.show()
```

Result: (different kernel size)





As a result, we can see the clearer edges of the circles in the eroded image, but cannot see the edge between the circles in the dilated image.

Opening

Opening operation is eroding the image and then dilating. According to these processes, it removes noise first which reduces the edge size by erosion, and then expands the edge of image by dilation. Therefore, it is used for removing the unwanted edge.

Closing

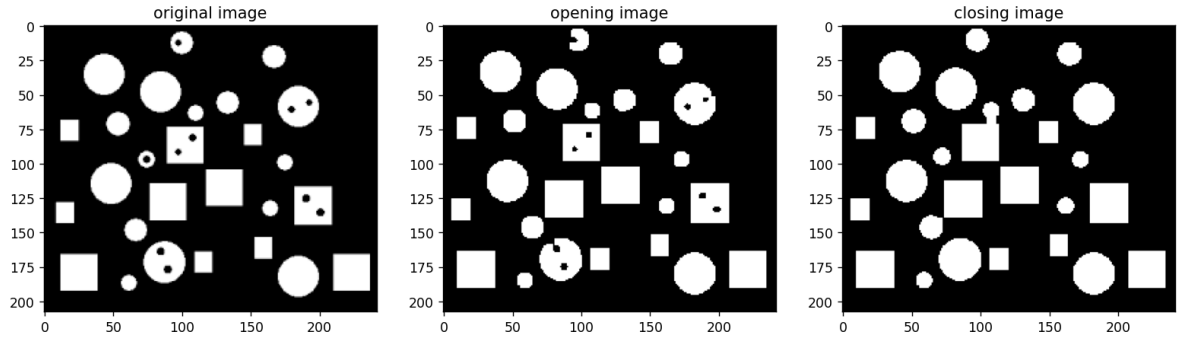
Closing is the opposite of opening. The image is dilated first, and then eroded. Therefore, the holes in the image will be filled by dilation and the thickness of the edge will be reduced by erosion in the final.

Coding:

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

img = cv2.imread('objects.png', 0)
(thresh, img2) = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY)
img2 = img2/255     # normalize to binary 0,1 image

h,w = img.shape

# Erosion
def erosion(img,h,w,kernel):
    s_element = np.ones((kernel,kernel))
    img_erosion= np.zeros((h,w))
    x  = int((kernel-1)/2)

    for i in range(x, h-x):
        for j in range(x, w-x):
            temp = img[i-x:i+x+1, j-x:j+x+1]
            product = temp*s_element
            img_erosion[i-1,j-1]= np.min(product)
    return img_erosion
```

```python
# Dilation
def dilation(img,h,w,kernel):
    s_element = np.ones((kernel,kernel))
    img_dilation= np.zeros((h,w))
    x  = int((kernel-1)/2)

    for i in range(x, h-x):
        for j in range(x, w-x):
            temp = img[i-x:i+x+1, j-x:j+x+1]
            product = temp*s_element
            img_dilation[i-1,j-1]= np.max(product)
    return img_dilation

# Opening
first_open = erosion(img2,h,w,5)
second_open = dilation(first_open,h,w,5)

# Closing
first_close = dilation(img2,h,w,5)
second_close = erosion(first_close,h,w,5)

plt.subplot(131)
plt.title('original image')
plt.imshow(img,cmap="gray")

plt.subplot(132)
plt.title('opening image')
plt.imshow(second_open,cmap='gray')

plt.subplot(133)
plt.title('closing image')
plt.imshow(second_close,cmap='gray')

plt.show()
```

Result:



As a result, we can see that the noise or some circles in the opening image disappear by erosion and cannot get back from dilation. Moreover the shape of each is quite not perfect. On the other hand, the closing image can fill the holes in the objects by the dilation and the size is quite the same as the original image by the erosion. However, the size of the kernel is important to make the different results.