R13922136 周洛君

Computer Vision HW5: Report

Main code structure:

In the source code file, the code contains two main parts:

1. The functions for requirement (a) to (d)

   (a) Dilation

   (b) Erosion

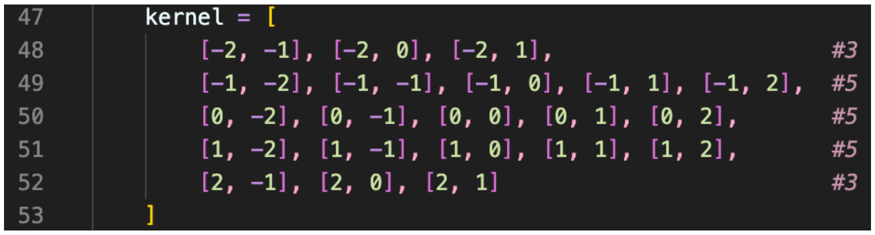
   (c) Opening

   (d) Closing

2. The main function, where the following things are done:

   2.1 read the gray scale lena.bmp image (this is done by cv2.imread())

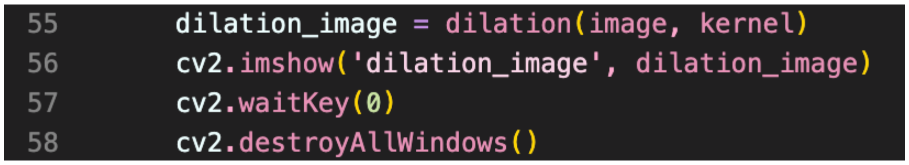
   2.2 define the octogonal 3-5-5-5-3 kernel



```python
    kernel = [
        [-2, -1], [-2, 0], [-2, 1],                          #3
        [-1, -2], [-1, -1], [-1, 0], [-1, 1], [-1, 2],       #5
        [0, -2], [0, -1], [0, 0], [0, 1], [0, 2],            #5
        [1, -2], [1, -1], [1, 0], [1, 1], [1, 2],            #5
        [2, -1], [2, 0], [2, 1]                              #3
    ]
```

(each element in the kernel is a coordinate, we will use these coordinates to perform translation in function (a) to (d))

   2.3 call the functions and show the resulting image of each morphological operation



```python
    dilation_image = dilation(image, kernel)
    cv2.imshow('dilation_image', dilation_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

(This is the part for (a) dilation, the other parts are almost the same, except that we modify the names, and call the corresponding function. For other operations (b) to (d) please refer the source code.)

Function Explanations:

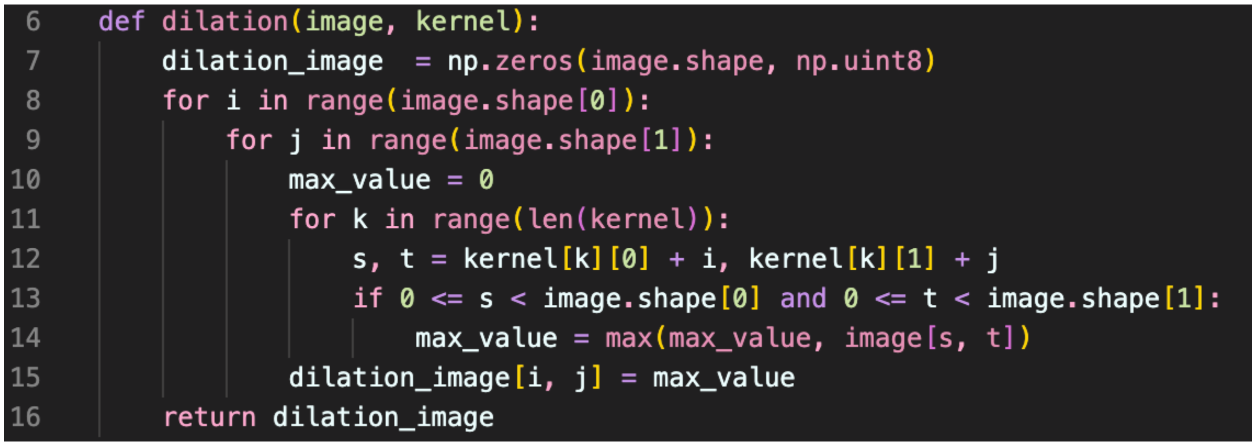
Dilation

The code for performing dilation on gray scale image is a little bit similar to what we have done for binary images, for example, the initialization part is the same, which is done in line 7, and in line 8-9, we loop through the pixel coordinates in the image.

The difference occurs from line 10, where we first initialize a variable “max_value” as 0, and this value is used to record the maximum value of the neighbors of each particular pixel in the image.

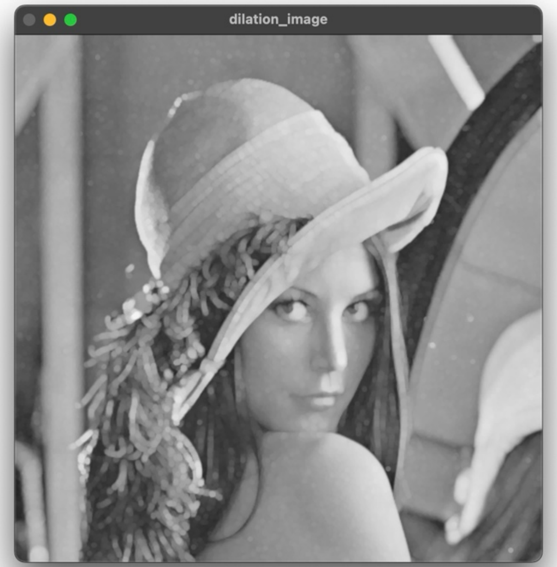
As I mentioned in the comments, grayscale dilation is done by assigning each pixel in the image with the maximum value of the pixels covered by the kernel, so after we calculated the translated coordinates in s, t, we then check if they are in the range of the image’s coordinates, if so, we find the maximum value by comparing this neighboring pixel’s value to the current maximum value, and in line 15, we save the ultimate maximum value after looping over all elements in the kernel.

More details are written in the comments.



```
 6  def dilation(image, kernel):
 7      dilation_image  = np.zeros(image.shape, np.uint8)
 8      for i in range(image.shape[0]):
 9          for j in range(image.shape[1]):
10              max_value = 0
11              for k in range(len(kernel)):
12                  s, t = kernel[k][0] + i, kernel[k][1] + j
13                  if 0 <= s < image.shape[0] and 0 <= t < image.shape[1]:
14                      max_value = max(max_value, image[s, t])
15              dilation_image[i, j] = max_value
16      return dilation_image
```

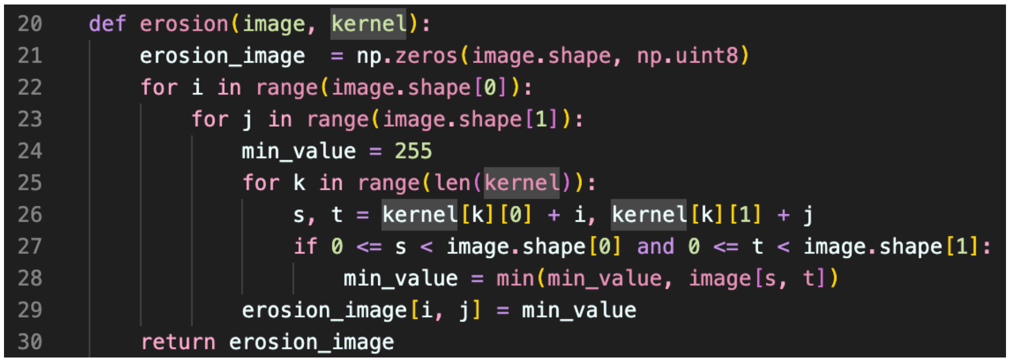
The resulting image after performing grayscale dilation:



Erosion

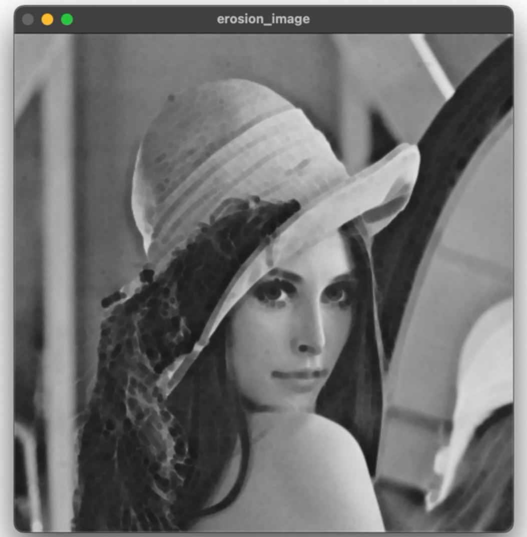
Grayscale erosion is defined similarly to grayscale dilation, except that we choose the minimum value in the neighbor, so we can see that most of the erosion function is the same as the dilation function, except that:

- We set min_value instead of max_value, and initialize it by 255 (line 24)
- Compare the translated coordinate (s, t)'s value with the current minimum value and update min_value (line 29)



```
20  def erosion(image, kernel):
21      erosion_image  = np.zeros(image.shape, np.uint8)
22      for i in range(image.shape[0]):
23          for j in range(image.shape[1]):
24              min_value = 255
25              for k in range(len(kernel)):
26                  s, t = kernel[k][0] + i, kernel[k][1] + j
27                  if 0 <= s < image.shape[0] and 0 <= t < image.shape[1]:
28                      min_value = min(min_value, image[s, t])
29              erosion_image[i, j] = min_value
30      return erosion_image
```

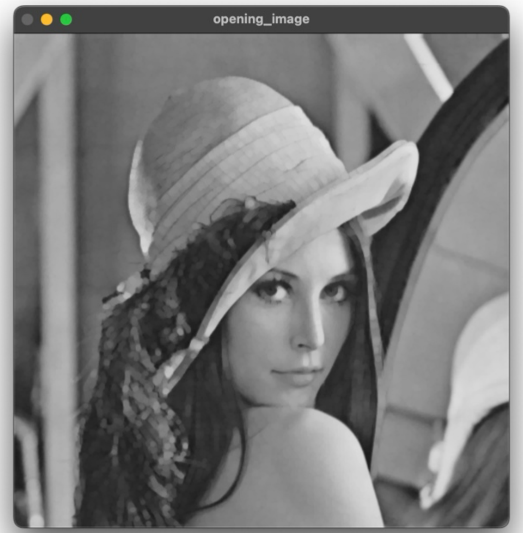
More details are also written in the comments, but they're really similar to the comments in dilation. The resulting picture of erosion:

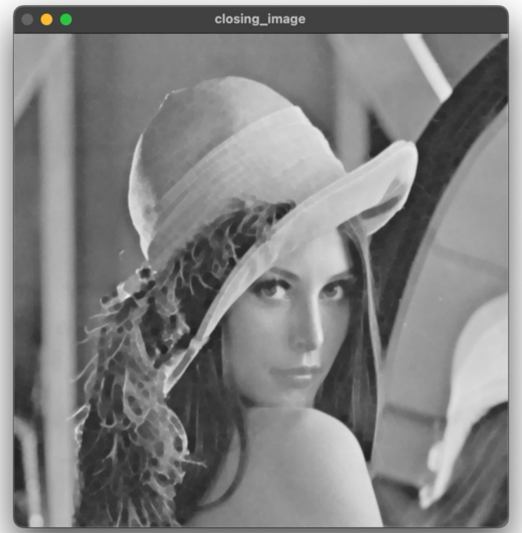


Opening / Closing

Opening and closing for grayscale images are defined the same as for binary images, which means:

- Opening = erosion followed by dilation
- Closing = dilation followed by erosion


opening_image


closing_image