922 U0610 電腦視覺 Computer Vision

Homework 5

授課教師: 傳楸善 教授

學生系級: 查工所一年級

學生姓名: _____姚嘉昇______

學生學號: <u>R06922002</u>

I. INTRODUCTION

1.1. Descriptions of Problem

This homework is to do gray scaled morphology with following rules:

- A. Please use the octagonal 3-5-5-3 kernel.
- B. Please take the local maxima or local minima respectively.
- C. Four images should be included in your report: Dilation, Erosion, Opening, and Closing.

1.2. Programming Tools

- 1.2.1. Programming Language: Python3
- 1.2.2. Programming IDE: Visual Studio Code

II. METHOD

- 2.1. Algorithms
- 2.1.1. Grayscaled Dilation

$$(f\oplus b)(x)=\sup_{y\in E}[f(y)+b(x-y)]$$

2.1.2. Grayscaled Erosion

$$(f\ominus b)(x)=\inf_{y\in B}[f(x+y)-b(y)]$$

2.1.3. Opening

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

2.1.4. Closing

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

2.2. Code Fragments

2.2.1. Code fragments of dilation

```
if __name__ == '__main__':
    from PIL import Image
    import numpy as np
    import JasonDIP
    # Define kernel for dilation.
    kernel = np.array([\
        [0, 1, 1, 1, 0], \
        [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [0, 1, 1, 1, 0]])
    centerKernel = (2, 2)
    originalImage = Image.open('lena.bmp')
    # Get dilation image.
    dilationImage = JasonDIP.dilation(originalImage, kernel, centerKernel)
    # Save image fo file.
    dilationImage.save('dilation.bmp')
```

Figure 2.2.1.1. Code of main of dilation.

```
def dilation(originalImage, kernel, centerKernel):
    :type originalImage: Image (from PIL)
    :type kernel: numpy array
    :type centerKernel: tuple
    :return type: Image (from PIL)
    from PIL import Image
   dilationImage = Image.new('L', originalImage.size)
   for r in range(originalImage.size[0]):
        for c in range(originalImage.size[1]):
           localMaxPixel = 0
            for x in range(kernel.shape[0]):
                for y in range(kernel.shape[1]):
                    if (kernel[x, y] == 1):
                        destX = r + (x - centerKernel[0])
                        destY = c + (y - centerKernel[1])
                        if ((0 <= destX < originalImage.size[0]) and \
                            (0 <= destY < originalImage.size[1])):</pre>
                            # Get pixel value in original image at (destX, destY)
                            originalPixel = originalImage.getpixel((destX, destY))
                            localMaxPixel = max(localMaxPixel, originalPixel)
           dilationImage.putpixel((r, c), localMaxPixel)
    return dilationImage
```

Figure 2.2.1.2. Code of function of dilation.

2.2.2. Code fragments of erosion

```
if __name__ == '__main__':
    from PIL import Image
    import numpy as np
    import JasonDIP
    kernel = np.array([\
        [0, 1, 1, 1, 0], \
        [1, 1, 1, 1, 1], \
       [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [0, 1, 1, 1, 0]])
    centerKernel = (2, 2)
   # Load image from file.
   originalImage = Image.open('lena.bmp')
    # Get erosion image.
   erosionImage = JasonDIP.erosion(originalImage, kernel, centerKernel)
   # Save image fo file.
    erosionImage.save('erosion.bmp')
```

Figure 2.2.2.1. Code of main of erosion.

```
def erosion(originalImage, kernel, centerKernel):
    :type originalImage: Image (from PIL)
    :type kernel: numpy array
    :type centerKernel: tuple
    :return type: Image (from PIL)
    from PIL import Image
    # New image with the same size and 'grayscale' format.
    erosionImage = Image.new('L', originalImage.size)
    for r in range(originalImage.size[0]):
        for c in range(originalImage.size[1]):
            localMinPixel = 255
            # Scan each column in kernel.
            for x in range(kernel.shape[0]):
                for y in range(kernel.shape[1]):
                    if (kernel[x, y] == 1):
                        destX = r + (x - centerKernel[0])
                        destY = c + (y - centerKernel[1])
                        if ((0 <= destX < originalImage.size[0]) and \
                            (0 <= destY < originalImage.size[1])):</pre>
                            originalPixel = originalImage.getpixel((destX, destY))
                            localMinPixel = min(localMinPixel, originalPixel)
            erosionImage.putpixel((r, c), localMinPixel)
    return erosionImage
```

Figure 2.2.2.2. Code of function of erosion.

2.2.3. Code fragments of opening

```
if name == ' main ':
   from PIL import Image
   import numpy as np
   import JasonDIP
   # Define kernel for opening.
   kernel = np.array([\
       [0, 1, 1, 1, 0], \
       [1, 1, 1, 1, 1], \
       [1, 1, 1, 1, 1], \
       [1, 1, 1, 1, 1], \
       [0, 1, 1, 1, 0]])
   # Define center of kernel for opening.
   centerKernel = (2, 2)
   originalImage = Image.open('lena.bmp')
   openingImage = JasonDIP.opening(originalImage, kernel, centerKernel)
   openingImage.save('opening.bmp')
```

Figure 2.2.3.1. Code of main of opening.

```
def opening(originalImage, kernel, centerKernel):

"""

type originalImage: Image (from PIL)

type kernel: numpy array

type centerKernel: tuple

return type: Image (from PIL)

"""

return dilation(erosion(originalImage, kernel, centerKernel), kernel, centerKernel)
```

Figure 2.2.3.2. Code of function of opening.

2.2.4. Code fragments of closing

```
if name == ' main ':
    from PIL import Image
   import numpy as np
    import JasonDIP
   # Define kernel for closing.
   kernel = np.array([\
       [0, 1, 1, 1, 0], \
       [1, 1, 1, 1, 1], \
       [1, 1, 1, 1, 1], \
       [1, 1, 1, 1, 1], \
       [0, 1, 1, 1, 0]])
   # Define center of kernel for closing.
   centerKernel = (2, 2)
   originalImage = Image.open('lena.bmp')
   # Get closing image.
   closingImage = JasonDIP.closing(originalImage, kernel, centerKernel)
    closingImage.save('closing.bmp')
```

Figure 2.2.4.1. Code of main of closing.

```
def closing(originalImage, kernel, centerKernel):
    """

stype originalImage: Image (from PIL)
    :type kernel: numpy array
    :type centerKernel: tuple
    :return type: Image (from PIL)
    """

return erosion(dilation(originalImage, kernel, centerKernel), kernel, centerKernel)
```

Figure 2.2.4.2. Code of function of closing.

III. RESULTS

3.1. Original Image



Figure 3.1. Original lena.bmp.

3.2. Results of dilation and erosion



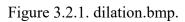




Figure 3.2.2. erosion.bmp.

3.3. Results of opening and closing



Figure 3.3.1. Original lena.bmp.



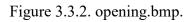




Figure 3.3.3. closing.bmp.