Homework 5

Mathematical Morphology - Gray Scaled Morphology

R09922063 鄭筠庭 資工所碩一

Programming language: Python 3.7.3

Library used for this homework:

- ✦ Numpy: for array usage
- ✦ Opencv: to read and write the image file

Image info: lena.bmp [512(width),512(height),1(Grayscale)]

Code explanation:

The kernel for this homework is an octogonal 3-5-5-5-3 kernel (taking the local maxima or local minima respectively).

“Grayscale morphology” is simply a generalization from 1 bpp (binary) images to images with multiple bits/pixel, where the Max and Min operations are used in place of the OR and AND.

(a) Dilation

Go through each pixel and find the local maximum within the octagonal kernel. Assign the local maximum to the new image. Need to check if the row number and column number is between 0 to 255.

```
for r in range(rows):
    for c in range(columns):
        localMax = 0
        #Go through a 5*5 kernel
        for numRow in range(len(inputKernel)):
            if (r+numRow-centerR) >= 0 and (r+numRow-centerR) < rows: #Check row isn't below 0 or larger than 512
                for numCol in range(len(inputKernel[0])):
                    if (c+numCol-centerC) >= 0 and (c+numCol-centerC) < columns: #Check column isn't below 0 or larger t
                            if inputKernel[numRow, numCol] == 1 and lena[r+numRow-centerR, c+numCol-centerC] > localMax:
                                localMax = lena[r+numRow-centerR, c+numCol-centerC]

        lenaCopy[r,c] = localMax
```

(b) Erosion

Go through each pixel and find the local minimum within the octagonal kernel. Assign the local minimum to the new image. Need to check if the row number and column number is between 0 to 255.

```
for r in range(rows):
    for c in range(columns):
        localMin = 255
        #Go through a 5*5 kernel
        for numRow in range(len(inputKernel)):
            if (r+numRow-centerR) >= 0 and (r+numRow-centerR) < rows: #Check row isn't below 0 or larger than 512
                for numCol in range(len(inputKernel[0])):
                    if (c+numCol-centerC) >= 0 and (c+numCol-centerC) < columns: #Check column isn't below 0 or larger th
                            if inputKernel[numRow, numCol] == 1 and lena[r+numRow-centerR, c+numCol-centerC] < localMin:
                                localMin = lena[r+numRow-centerR, c+numCol-centerC]

        lenaCopy[r,c] = localMin
```

(c) Opening

First execute erosion on the Lena image, then execute dilation on the eroded image.

```
def opening(lena, inputKernel):
    lenaCopy = np.zeros(lena.shape, np.uint8)
    lenaCopy = lena.copy()

    lenaCopy = erosion(lena,inputKernel)
    cv2.imwrite("opening_gray.bmp", dilation(lenaCopy, inputKernel))
```

(d) Closing

First execute dilation on the Lena image, then execute erosion on the processed image.

```
def closing(lena, inputKernel):
    lenaCopy = np.zeros(lena.shape, np.uint8)
    lenaCopy = lena.copy()

    lenaCopy = dilation(lena,inputKernel)
    cv2.imwrite("closing_gray.bmp", erosion(lenaCopy, inputKernel))
```

Result:

(a) Dilation

(b) Erosion

(c) Opening

(d) Closing