Homework 4

Mathematical Morphology - Binary Morphology

R09922063 鄭筠庭 資工所碩一

Programming language: Python 3.7.3

Library used for this homework:

- ✦ Numpy
- ✦ OpenCv: to read and write the image file

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

Code explanation:

First, read in lena.bmp as grayscale image and turn it into a binary image.

```
def binarize():
    for r in range(rows):
        for c in range(columns):
            if lena[r,c] < 128:
                lena[r,c] = 0
            else:
                lena[r,c] = 255
```

(a)Document about dilation: https://en.wikipedia.org/wiki/Dilation_(morphology)

Go through each pixel with two “for” loop. If the pixel is black (0), skip it. Next step is to go through a 5*5 kernel. Check if the corresponding pixels on the origin image and copied image are both black. If they are black, change pixel on the copied image to the same one on the kernel.

```
if lena[r,c] == 0:
    continue

#Go through a 5*5 kernel
for numRow in range(0,len(inputKernel)):
    if (r+numRow-centerR) >= 0 and (r+numRow-centerR) < rows: #Check row is
        for numCol in range(0,len(inputKernel)):
            if (c+numCol-centerC) >= 0 and (c+numCol-centerC) < columns: #C
                if inputKernel[numRow, numCol] == 255: #Check if the po
                    lenaCopy[r+numRow-centerR, c+numCol-centerC] = 255
```

(b)Document about erosion: https://en.wikipedia.org/wiki/Erosion_(morphology)

The beginning part is same as dilation. Go through each pixel with two “for” loop and DON’T skip the black pixel. Use a Boolean variable “flagSame” to record the current state of erosion. If a mismatch is found, the flag will be captured as False, and the target pixel will be recorded as black.

```
for r in range(rows):
    for c in range(columns):
        flagSame = True
        for numRow in range(0,len(inputKernel)):
            if flagSame == False:
                break
            if (r+numRow-centerR) >= 0 and (r+numRow-centerR) < rows: #Check row
                for numCol in range(0,len(inputKernel)):
                    if (c+numCol-centerC) >= 0 and (c+numCol-centerC) < columns:
                        if inputKernel[numRow, numCol] == 255: #Check if the pos
                            if lena[r+numRow-centerR, c+numCol-centerC] != 255:
                                flagSame = False
                                break
```

(c) Document about opening: https://en.wikipedia.org/wiki/Opening_(morphology)

Call erosion function, and call dilation function using the erosion result.

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

```
lenaCopy = erosion(lena, lenaCopy, kernel)
lenaCopyCopy = np.zeros(lenaOrigin.shape, np.uint8)
lenaCopyCopy = lenaCopy.copy()
lenaCopy = dilation(lenaCopy,lenaCopyCopy, kernel)
```

(d) Document about closing: https://en.wikipedia.org/wiki/Closing_(morphology)

Call dilation function, and call erosion dilation function using the dilation result.

$A \bullet B = (A \oplus B) \ominus B,$

```
lenaCopy = dilation(lena,lenaCopy, kernel)
lenaCopyCopy = np.zeros(lenaOrigin.shape, np.uint8)
lenaCopyCopy = lenaCopy.copy()
lenaCopy = erosion(lenaCopy, lenaCopyCopy, kernel)
```

(e)Document: https://en.wikipedia.org/wiki/Hit-or-miss_transform

The main formula for calculating hit-and-miss: $A \odot B = (A \ominus C) \cap (A^c \ominus D)$

Call the erosion function with the 3*3 kernel and original Lena image, and call the erosion function with the disjoint kernel and complement Lena image. Find the intersection of this two sets.

```
lenaCopy = erosion(lena, lenaCopy, kernel_J)

# Reverse the image and make a copy of it
lena = complement(lena)
lenaReverseCopy = np.zeros(lenaOrigin.shape, np.uint8)
lenaReverseCopy = lena.copy()

# Erosion for the reversed image and disjoint kernel
lenaReverseCopy = erosion(lena,lenaReverseCopy, disjointKernel)
```

Result:

(a) Dilation

(b) Erosion

(c) Opening

(d) Closing

(e) Hit-and-miss