電腦視覺 Homework 4 – Mathematical Morphology - Binary Morphology

資工碩一 r08922a04 林承德

Language : python 3.7

Library :

```
import numpy as np
from PIL import Image, ImageDraw
```

Execution way: python3 hw4.py (please put lena.bmp at the same directory as hw3.py)

Problems:

Problem(a)Dilation

Result

Code

```
def dilation(self, bin_image, kernel):
    dilation_image = Image.new('1', bin_image.size)
    h, w = bin_image.size
    pixels = bin_image.load()
    new_pixels = dilation_image.load()
    for i in range(h):
        for j in range(w):
            for point in kernel:
                new_i = i + point[0]
                new_j = j + point[1]
                if new_i >= 0 and new_i < h and new_j >= 0 and new_j < w:
                    if pixels[new_i, new_j] == 1:
                        new_pixels[i, j] = 1
                        break
    return dilation_image
```

Description:

After binarization the origin image, I new a new image initial with 0 which have the same size with the origin image. Using two layer for loop to go through all pixel and check it's neighbor by given kernel, if the current pixel could match the kernel, than the same pixel in output image should set to 1.

Problem(b) Erosion

Result

Code

```
def erosion(self, bin_image, kernel):
    erosion_image = Image.new('1', bin_image.size, color = 1)
    h, w = bin_image.size
    pixels = bin_image.load()
    new_pixels = erosion_image.load()
    for i in range(h):
        for j in range(w):
            for point in kernel:
                new_i = i + point[0]
                new_j = j + point[1]
                if new_i >= 0 and new_i < h and new_j >= 0 and new_j < w:
                    if pixels[new_i, new_j] != 1:
                        new_pixels[i, j] = 0
                        break
    return erosion_image
```

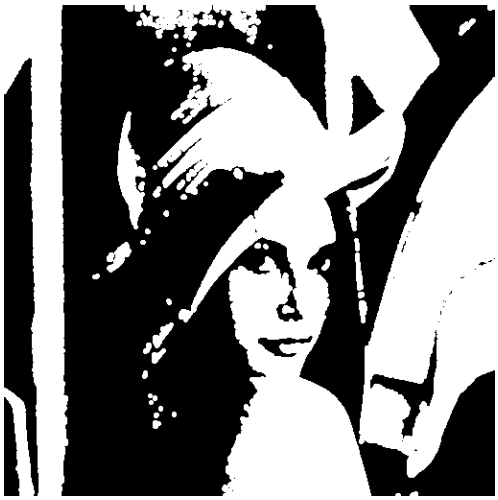
Description

this case is really similar to the previous one, I still binarization the origin image, new an output image with init value = 1 and have the same size with origin one. After this, I use two layer for loop to go though all pixels and

check every pixel, whether it's neighbor match the kernel. If no, then the same place pixel in output image should set to 0.

Problem(c) Opening

Result



Code

```
def opening(self, bin_image):
    erosion_image = self.erosion(bin_image, self.kernel)
    opening_image = self.dilation(erosion_image, self.kernel)
    return opening_image
```

Description

After binarization the origin image, I first did the erosion action, then done the dilation action, then the result would be opening.

Problem(d) Closing

Result

Code

```
def closing(self, bin_image):
    dilation_image = self.dilation(bin_image, self.kernel)
    closing_image = self.erosion(dilation_image, self.kernel)
    return closing_image
```

Description

After binarization the origin image, I first did the dilation action, then done the erosion action, then the result would be closing.

Problem(e) Hit and Miss transform

Result

Code

```
def complement(sefl, bin_image):
    complement_image = Image.new('1', bin_image.size)
    complement_pixels = complement_image.load()
    binary_pixels = bin_image.load()
    h, w = bin_image.size
    for i in range(h):
        for j in range(w):
            complement_pixels[i,j] = 1 - binary_pixels[i,j]
    return complement_image

def intersection(self, image1, image2):
    pixels1 = image1.load()
    pixels2 = image2.load()
    intersect_image = Image.new('1', image1.size)
    intersect_pixels = intersect_image.load()
    h, w = intersect_image.size
    for i in range(h):
        for j in range(w):
            intersect_pixels[i, j] = pixels1[i, j] & pixels2[i, j]
    return intersect_image

def hitandmiss(self, bin_image):
    complement_image = self.complement(bin_image)
    pixels = bin_image.load()
    hit = self.erosion(bin_image, self.ker_j)
    miss = self.erosion(complement_image, self.ker_k)
    return self.intersection(hit, miss)
```

Description

I binarization the origin image and call it binary_image, then make a complement image of binary_image which named complement_image. Do erosion on binary_image with kernel J then do erosion on complement image with kernel K, finally do and operation on the two previous result.