CV_HW4

R07922125羅偉倫

Language and Tool

- Python3.6
- Numpy
- PIL

Kernel

```
kernel = [      [-2, -1], [-2, 0], [-2, 1],
          [-1, -2], [-1, -1], [-1, 0], [-1, 1], [-1, 2],
          [0, -2], [0, -1], [0, 0], [0, 1], [0, 2],
          [1, -2], [1, -1], [1, 0], [1, 1], [1, 2],
                   [2, -1], [2, 0], [2, 1]]
kernel_J = [[1,0], [0,0],[0,-1]]
kernel_K = [[0,1],[-1,1],[-1,0]]
```

Input and binarize lena picture

```
# origin
img = Image.open('lena.bmp')
img_array = np.array(img)
Image.fromarray(img_array).save('lena.jpg')

# binary
print("Binarize the origin picture..\n")
img_b_array = np.uint8(np.copy(img_array)>=128)*255
Image.fromarray(img_b_array).save('lena_bin_128.jpg')
```

lena

lena_bin_128

Problem1 Dilation

I trace all the pixels of the binary picture, and make the pixel dilated by the kernel.

img_array:numpy array,the array of a picture
kernel:list,the kernel I use
pixel:int,the value of pixel we process

img_b_array:array,the array of the binary picture

```
def dilation(img_array,kernel,pixel):
    img_d = np.copy(img_array)
    for i in range(img_array.shape[0]):
        for j in range(img_array.shape[1]):
            if(img_array[i][j]==pixel):
                for k in kernel:
                    new_x = i + k[0]
                    new_y = j + k[1]
                    if(new_x >=0 and new_x<img_array.shape[0] and
new_y>=0 and new_y<img_array.shape[1]):
                        img_d[new_x][new_y] = pixel
    return img_d
img_d = dilation(img_b_array,kernel,255)
Image.fromarray(img_d).save('lena_bin_dil.jpg')
```

lena_dil

Problem2 Erosion

During tracing all pixels of the binary picture,I count how many kernel pixels value in the binary picture is ‘pixel’.If the counting number is the same as the length of kernel, the pixel value will be ‘pixel’.

img_array:numpy array,the array of the binary picture
kernel:list,the kernel I use
pixel:int,the value of pixel we process

img_b_array:array,the array of the binary picture

```
def erosion(img_array,kernel,pixel):
    img_e = np.copy(img_array)
    length = len(kernel)
    for i in range(img_array.shape[0]):
        for j in range(img_array.shape[1]):
            count = 0
            for k in kernel:
                new_x = i + k[0]
                new_y = j + k[1]
                if(new_x >=0 and new_x<img_array.shape[0] and new_y>=0
and new_y<img_array.shape[1] and img_array[new_x][new_y]==pixel):
                    count += 1
            if(count < length):
                img_e[i][j] = 255-pixel
            if(count == length):
                img_e[i][j] = pixel
    return img_e
img_e = erosion(img_b_array,kernel,255)
Image.fromarray(img_e).save(‘lena_bin_ero.jpg')
```

lena_ero

Problem 3&4 Opening and Closing

To perform opening on the binary picture,I do erosion first and then do dilation.
To perform closing on the binary picture,I do dilation first and then do erosion.

img_array:numpy array,the array of the binary picture
kernel:list,the kernel I use
pixel:int,the value of pixel we process

img_b_array:array,the array of the binary picture

```
def opening(img_array,kernel,pixel):
    return dilation(erosion(img_array,kernel,pixel),kernel,pixel)
img_open = opening(img_b_array,kernel,255)
Image.fromarray(img_open).save('lena_bin_open.jpg')

def closing(img_array,kernel,pixel):
    return erosion(dilation(img_array,kernel,pixel),kernel,pixel)
img_close = closing(img_b_array,kernel,255)
Image.fromarray(img_close).save('lena_bin_close.jpg')
```

lena_open

lena_close

Problem5 Hit-and-Miss

First,I get the erosions of the binary picture by kernel_J and kernel_K on white and black pixels separately.
Second,I do the union operation on the two array above.

img_array:numpy array,the array of the binary picture
kernel_J,kernel_K:list,the kernel I use
pixel:int,the value of pixel we process

img_b_array:array,the array of the binary picture

```
def hit_and_miss(img_array,kernel_J,kernel_K,pixel):
    img_e_positive = erosion(img_array,kernel_J,pixel)
    img_e_negative = erosion(img_array,kernel_K,255-pixel)
    img_h_a_m = np.copy(img_e_positive)

    for i in range(img_array.shape[0]):
        for j in range(img_array.shape[1]):
            if(img_h_a_m[i][j] == pixel):
                if(img_e_negative[i][j] != 255-pixel):
                    img_h_a_m[i][j] = 255-pixel
    return img_h_a_m
img_h_a_m = hit_and_miss(img_b_array,kernel_J,kernel_K,255)
Image.fromarray(img_h_a_m).save('lena_bin_ham.jpg')
```

lena_hit_and_miss