R07922141
張緣彩
19/10/2018

Computer Vision Hw4 Report

I. Implementation

a. Dilation

As described in the lecture note, dilation actually makes a white pixel (255) wider. For given center point and a kernel, if the center point is 255 then all other neighbors point (according to kernel) will be 255 as well.

```
def dilation(img, kernel):
    to_return = img.copy()
    for i in range(512):
        for j in range(512):
            if img[i][j] == 255:
                for k in kernel:
                    if (i+k[0] >= 0 and i+k[0] <=511) and (j+k[1] >= 0  and j+k[1] <= 511):
                        to_return[i+k[0]][j+k[1]] = 255
    return to_return

dia_lena = dilation(bi_lena, get_neighbour((2,2), kernel01))
cv2.imshow('Dilation lena', dia_lena)
```

b. Erosion

As described in the lecture note, erosion actually makes a black pixel (0) wider. For given center point and a kernel, the center point will be 255 if and only if all the neighbors point of the center point is 255, otherwise the center point will be 0.

```
def erosion(img, kernel):
    to_return = img.copy()
    for i in range(512):
        for j in range(512):
            flag = True
            for k in kernel:
                if (i+k[0] >= 0 and i+k[0] <=511) and (j+k[1] >= 0  and j+k[1] <= 511):
                    if img[i+k[0]][j+k[1]] != 255:
                        flag = False
                        break
                else:
                    flag = False
                    break
            if flag:
                to_return[i][j] = 255
            else:
                to_return[i][j] = 0

    return to_return

ero_lena = erosion(bi_lena, get_neighbour((2,2), kernel01))
cv2.imshow('Erosion lena', ero_lena)
```

c. Opening

As described in the lecture note, we just need to apply erosion to the given image and continue by dilation.

```
def opening(img, kernel):
    to_return = erosion(img, kernel)
    to_return = dilation(to_return, kernel)
    return to_return

open_lena = opening(bi_lena, get_neighbour((2,2), kernel01))
cv2.imshow('Opening lena', open_lena)
```

d. Closing

As described in the lecture note, we just need to apply dilation to the given image and continue by erosion.

```
def closing(img, kernel):
    to_return = dilation(img, kernel)
    to_return = erosion(to_return, kernel)
    return to_return

close_lena = closing(bi_lena, get_neighbour((2,2), kernel01))
cv2.imshow('Closing lena', close_lena)
```

e. Hit and Miss

As described in the lecture note, we need to calculate erosion of A and J kernel, and erosion of A complement and K kernel. Then intersect these two results to get the final result.

```
def hit_miss(img, kernel_pattern):
    reversed_img = img.copy()

    for i in range(512):
        for j in range(512):
            if reversed_img[i][j] == 255:
                reversed_img[i][j] = 0
            else:
                reversed_img[i][j] = 255

    Jkernel = get_neighbour((0,1), kernel_pattern)
    Kkernel = get_neighbour((1,0), kernel_pattern)

    A = erosion(img, Jkernel)
    Ac = erosion(reversed_img, Kkernel)

    to_return = img.copy()

    for i in range(512):
        for j in range(512):
            if A[i][j] == 255 and Ac[i][j] == 255:
                to_return[i][j] = 255
            else:
                to_return[i][j] = 0

    return to_return

hit_miss_lena = hit_miss(bi_lena, kernel02)
cv2.imshow('Hit and miss lena', hit_miss_lena)
```

f. Kernel

This function is use to get all the point in the given kernel. For given a center point and kernel pattern, it generates all the neighbors point according to the center point.

```
# Get neighbor point according to given center
def get_neighbour(center, kernel):
    to_return = []
    for y in range(len(kernel)):
        for x in range(len(kernel[0])):
            if kernel[x][y] == 1:
                to_return.append([x - center[0], y - center[1]])
    return to_return
```

II. Result

a. Dilation

b. Erosion

c. Opening

d. Closing

e. Hit and Miss