CV hw5 / 電機所R06921082 陳與賢

Description:

利用python來處理bmp檔，進行「灰階圖」的dilation、erosion、opening、 closing共4種操作。

（一）Dilation

- *Algorithm*

首先會把kernel存成與原點的相對距離，比方說octonal 3-5-5-5-3 kernel 會存成如下：

```
63 kernel = []
64 for i in range(5):
65     for j in range(5):
66         temp = [i-2, j-2]
67         kernel.append(temp)
68 kernel.remove([-2,-2])
69 kernel.remove([-2,2])
70 kernel.remove([2,-2])
71 kernel.remove([2,2])
72
```

再來先把輸出圖的pixels都填0(黑色)，開始跑dilation
因為這次作業的kernel的value都是0，所以在原圖上的pixels值若非0才需要檢查，首先先看這次有檢查的kernel所對應的點是否超出圖的大小，若沒有則比較pixel值是否比kernel所對應的點的output值還大，若是則替換pixel值過去，code 如下：

```
# initial
for x in range(dilation_output.width):
    for y in range(dilation_output.height):
        dilation_pixels[x, y] = 0

for x in range(img_input.width):
    for y in range(img_input.height):
        if pixels_input[x, y] != 0:
            for z in range(len(kernel)):
                x1 = x + kernel[z][0]
                y1 = y + kernel[z][1]
                if x1 < 0 or x1 > 511 or y1 < 0 or y1 > 511:
                    continue
                else:
                    if pixels_input[x, y] > dilation_pixels[x1, y1]:
                        dilation_pixels[x1, y1] = pixels_input[x, y]
```

- *Result*

（二）Erosion

- *Algorithm*

首先會把kernel存成與原點的相對距離，同dilation的做法。
再來開始跑erosion，檢查kernel是否超出範圍以及kernel對應到原圖位置的pixels值是否為0，若沒超出範圍且對應的點均不為0，則找出kernel所對應的pixel值的最小值(因為kernel的value=0所以不用減)，然後assign到現在在原圖上檢查的點，code如下：

```
for x in range(img_input.width):
    for y in range(img_input.height):
        flag = True
        min_pixel = 255
        # check boundry
        for z in range(len(kernel)):
            x1 = x + kernel[z][0]
            y1 = y + kernel[z][1]
            if x1 < 0 or x1 > 511 or y1 < 0 or y1 > 511:
                flag = False
                break
            if pixels_input[x1, y1] == 0:
                flag = False
                break
            if pixels_input[x1, y1] < min_pixel:
                min_pixel = pixels_input[x1, y1]
        if flag == True:
            erosion_pixels[x, y] = min_pixel
```

- *Result*

（三）Opening、Closing

- *Algorithm*

Opening就是先做erosion，發現侵蝕過頭了所以再做dilation
而Closing就事先做dilation，發現膨脹過頭了所以再做erosion
因為我erosion跟dilation都寫成function的型式，所以直接call即可
code如下：

```
73 output = dilation(kernel, "lena.bmp")
74 output.save("dilation.bmp")
75 output = erosion(kernel, "lena.bmp")
76 output.save("erosion.bmp")
77 output = dilation(kernel, "erosion.bmp")
78 output.save("opening.bmp")
79 output = erosion(kernel, "dilation.bmp")
80 output.save("closing.bmp")
```

- *Result*

opening

closing