

Computer Vision HW10

R06922075 翁瑋

- Write the following programs to detect edge:
Zero-crossing on the following four types of images to get edge images
(choose proper thresholds)
 1. Laplacian
 2. minimum-variance Laplacian
 3. Laplacian of Gaussian
 4. Difference of Gaussian, (use tk to generate D.O.G.) dog (inhibitory , excitatory , kernel size=11)

這一次的 edge detection 基本上的概念都一樣，都是使用 convolution，只是當中的 mask 選擇不同

```
3 def Laplace1(im, threshold):
4     pixels = im.load()
5     im_Laplace1 = Image.new('L', (im.size[0]-2, im.size[1]-2), 'black')
6     masks = [[0,1,0], [1,-4,1], [0,1,0]]
7
8     for i in range(im_Laplace1.size[0]):
9         for j in range(im_Laplace1.size[1]):
10             result = 0
11             for m in range(3):
12                 for n in range(3):
13                     result += pixels[i+m,j+n] * masks[m][n]
14             result = 0 if result > threshold else 255
15             im_Laplace1.putpixel((i,j), result)
16
17     return im_Laplace1
```

第一個 Laplacian Mask

0	1	0
1	-4	1
0	1	0

```
19 def Laplace2(im, threshold):
20     pixels = im.load()
21     im_Laplace2 = Image.new('L', (im.size[0]-2, im.size[1]-2), 'black')
22     masks = [[1,1,1], [1,-8,1], [1,1,1]]
23
24     for i in range(im_Laplace2.size[0]):
25         for j in range(im_Laplace2.size[1]):
26             result = 0
27             for m in range(3):
28                 for n in range(3):
29                     result += pixels[i+m,j+n] * masks[m][n]
30             result = result//3
31             result = 0 if result > threshold else 255
32             im_Laplace2.putpixel((i,j), result)
33
34     return im_Laplace2
```

第二個 Laplacian Mask

1	1	1
1	-8	1
1	1	1

```

36 def Minimum_variance_Laplacian(im, threshold):
37     pixels = im.load()
38     im_MvL = Image.new('L', (im.size[0]-2, im.size[1]-2), 'black')
39     masks = [[2,-1,2], [-1,-4,-1], [2,-1,2]]
40
41     for i in range(im_MvL.size[0]):
42         for j in range(im_MvL.size[1]):
43             result = 0
44             for m in range(3):
45                 for n in range(3):
46                     result += pixels[i+m,j+n] * masks[2-m][2-n]
47             result = result//3
48             result = 0 if result > threshold else 255
49             im_MvL.putpixel((i,j), result)
50
51     return im_MvL

```

Minimum Variance Laplacian

2	-1	2
-1	-4	-1
2	-1	2

```

53 def Laplace_of_Gaussian(im, threshold):
54     pixels = im.load()
55     im_Log = Image.new('L', (im.size[0]-10, im.size[1]-10), 'black')
56     masks = [
57         [ 0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0],
58         [ 0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
59         [ 0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
60         [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
61         [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
62         [-2, -9, -23, -1, 103, 178, 103, -1, -23, -9, -2],
63         [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
64         [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
65         [ 0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
66         [ 0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
67         [ 0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0]]
68
69     for i in range(im_Log.size[0]):
70         for j in range(im_Log.size[1]):
71             result = 0
72             for m in range(11):
73                 for n in range(11):
74                     result += pixels[i+m,j+n] * masks[10-m][10-n]
75             result = 0 if result > threshold else 255
76             im_Log.putpixel((i,j), result)
77
78     return im_Log

```

Laplace Of Gaussian





```

80 def Difference_of_Gaussian(im, threshold):
81     pixels = im.load()
82     im_DoG = Image.new('L', (im.size[0]-10, im.size[1]-10), 'black')
83     masks = [
84         [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1],
85         [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
86         [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
87         [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
88         [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
89         [-8, -13, -17, 15, 160, 283, 160, 15, -17, -13, -8],
90         [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
91         [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
92         [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
93         [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
94         [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1]]
95
96     for i in range(im_DoG.size[0]):
97         for j in range(im_DoG.size[1]):
98             result = 0
99             for m in range(11):
100                 for n in range(11):
101                     result += pixels[i+m,j+n] * masks[10-m][10-n]
102             result = 0 if result < threshold else 255
103             im_DoG.putpixel((i,j), result)
104
105     return im_DoG

```

Difference Of Gaussian

Result :

原圖	第一個 Laplacian(15)
	
第二個 Laplacian(15)	Minimum Variance Laplacian(20)
	
Laplace Of Gaussian(3000)	Difference Of Gaussian(1)
