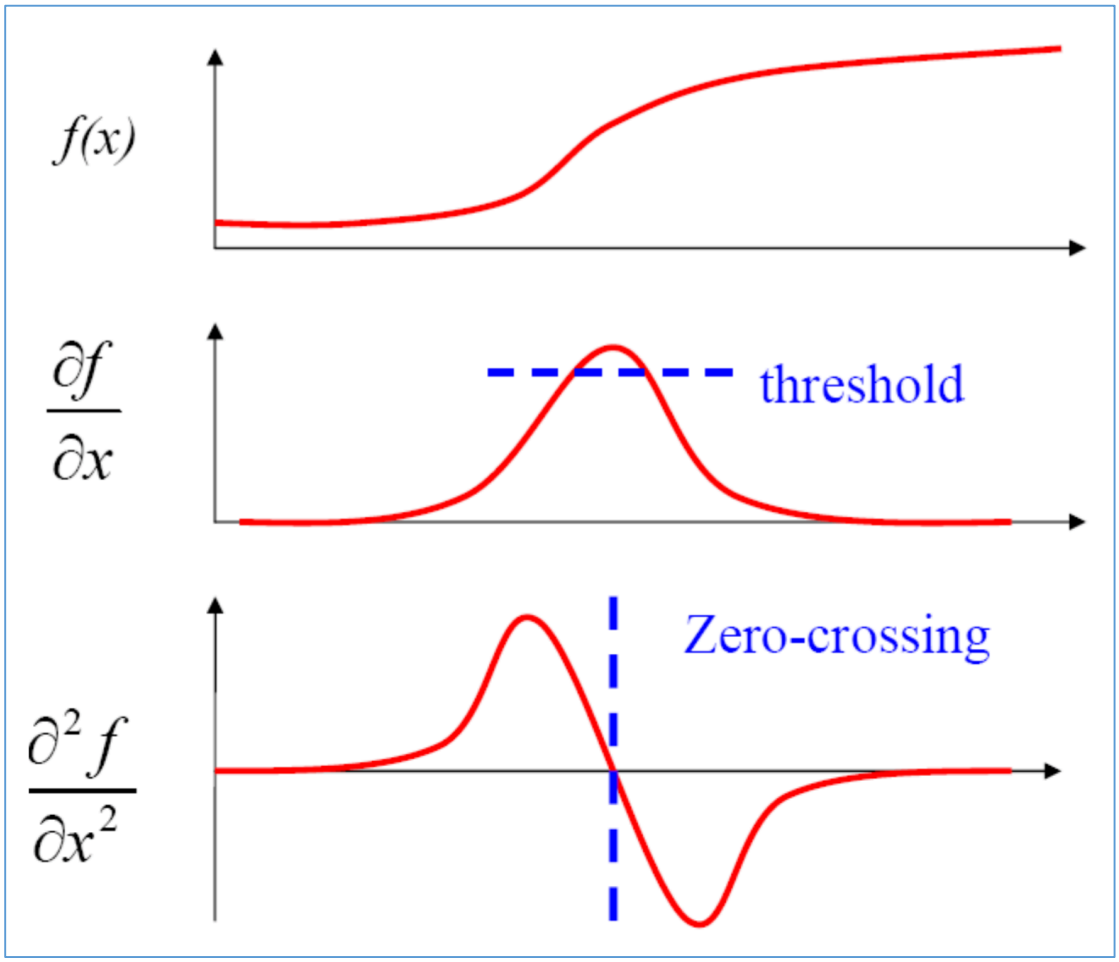
CV hw10 / 電機所 R06921082 陳與賢

Description:

利用 python 來處理 bmp 檔，使用各種 mask 以及 threshold 來做 edge detection，跟 hw9 的差別在於這次是 zero crossing 的版本，hw9 是用一階微分後找 pixel 值劇烈變化的地方，hw10 是用二階微分找到 pixel 值變化的轉折點，如下圖所示：

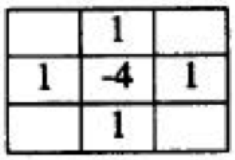


使用二階微分的優點在於，若邊界較寬，則 pixel 值變化和緩，那用一階微分就會難以區別何處才是真正的 edge，而二階微分可以較為準確的標出變化最劇烈的地方。

Algorithm:

@ Laplace Mask1

kernel:

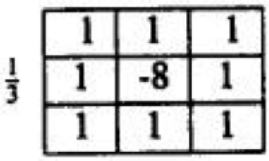


threshold: 20

將 input img 與此 kernel 做 cross-correlation，得到的值若大於 threshold 則 output 給值 0，反之給值 255

@ Laplace Mask2

kernel:

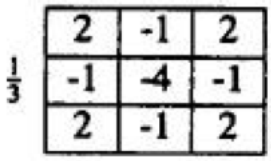


threshold: 20

將 input img 與此 kernel 做 cross-correlation，得到的值若大於 threshold 則 output 給值 0，反之給值 255

@ Minimum variance Laplacian

kernel:



threshold: 20

將 input img 與此 kernel 做 cross-correlation，得到的值若大於 threshold 則 output 給值 0，反之給值 255

@ Laplace of Gaussian

首先用下面公式算出 kernel

$$\triangle[G_\sigma(x,y) * f(x,y)] = [\triangle G_\sigma(x,y)] * f(x,y) = LoG * f(x,y)$$

$$LoG \triangleq \triangle G_\sigma(x,y) = \frac{\partial^2}{\partial x^2} G_\sigma(x,y) + \frac{\partial^2}{\partial y^2} G_\sigma(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2}$$

code 如下：

```
kernel = np.zeros((11, 11))
for i in range(11):
    out_tmp = ""
    for j in range(11):
        kernel[i, j] = LoG(i-5, j-5, 1)
```

```
def LoG(x, y, std):
    return (((x ** 2 + y ** 2 - 2 * (std ** 2)) / (std ** 4)) * (math.e ** ((-1 * (x ** 2 + y ** 2)) / (2 * (std ** 2)))))
```

sigma: 1

kernel:（因為排版 我沒有把全部小數點都印出來）

```
6.666213 4.875596 1.324780 1.361738 5.424790 8.571302 5.424790 1.361738 1.324780 4.875596 6.666213
4.875596 3.376055 8.571302 0.000817 0.003052 0.004696 0.003052 0.000817 8.571302 3.376055 4.875596
1.324780 8.571302 0.001974 0.016537 0.053903 0.077762 0.053903 0.016537 0.001974 8.571302 1.324780
1.361738 0.000817 0.016537 0.109893 0.246254 0.270670 0.246254 0.109893 0.016537 0.000817 1.361738
5.424790 0.003052 0.053903 0.246254 0.000000 -0.60653 0.000000 0.246254 0.053903 0.003052 5.424790
8.571302 0.004696 0.077762 0.270670 -0.60653 -2.00000 -0.60653 0.270670 0.077762 0.004696 8.571302
5.424790 0.003052 0.053903 0.246254 0.000000 -0.60653 0.000000 0.246254 0.053903 0.003052 5.424790
1.361738 0.000817 0.016537 0.109893 0.246254 0.270670 0.246254 0.109893 0.016537 0.000817 1.361738
1.324780 8.571302 0.001974 0.016537 0.053903 0.077762 0.053903 0.016537 0.001974 8.571302 1.324780
4.875596 3.376055 8.571302 0.000817 0.003052 0.004696 0.003052 0.000817 8.571302 3.376055 4.875596
6.666213 4.875596 1.324780 1.361738 5.424790 8.571302 5.424790 1.361738 1.324780 4.875596 6.666213
```

threshold: 30

將 input img 與此 kernel 做 cross-correlation，得到的值若大於 threshold 則 output 給值 0，反之給值 255

@ Difference of Gaussian

首先用下面公式算出 kernel

$$Gaussian: \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{r^2+c^2}{\sigma^2}\right)}$$

$$DoG \triangleq G_{\sigma_1} - G_{\sigma_2}$$

code 如下：
（因為用 G(1)-G(3)出來的值都會是負的，所以我都先乘-1 變正數）

```
kernel = np.zeros((11, 11))
for i in range(11):
    out_tmp = ""
    for j in range(11):
        kernel[i, j] = -1*DoG(i-5, j-5, 1, 3)
```

```
def gaussian(x, y, std):
    return (math.e ** ((-1 * (x ** 2 + y  ** 2)) / (2 * (std ** 2)))) / ((2 * math.pi * (std ** 2)))
def DoG(x, y, std1, std2):
    return gaussian(x, y, std1) - gaussian(x, y, std2)
```

sigma1: 1
sigma2: 3
kernel:（因為排版 我沒有把全部小數點都印出來）

```
0.001099 0.001812 0.002674 0.003530 0.004170 0.004408 0.004170 0.003530 0.002674 0.001812 0.001099
0.001812 0.002988 0.004408 0.005814 0.006844 0.007216 0.006844 0.005814 0.004408 0.002988 0.001812
0.002674 0.004408 0.006485 0.008349 0.009073 0.008957 0.009073 0.008349 0.006485 0.004408 0.002674
0.003530 0.005814 0.008349 0.008423 0.000330 -0.00737 0.000330 0.008423 0.008349 0.005814 0.003530
0.004170 0.006844 0.009073 0.000330 -0.04272 -0.07980 -0.04272 0.000330 0.009073 0.006844 0.004170
0.004408 0.007216 0.008957 -0.00737 -0.07980 -0.14147 -0.07980 -0.00737 0.008957 0.007216 0.004408
0.004170 0.006844 0.009073 0.000330 -0.04272 -0.07980 -0.04272 0.000330 0.009073 0.006844 0.004170
0.003530 0.005814 0.008349 0.008423 0.000330 -0.00737 0.000330 0.008423 0.008349 0.005814 0.003530
0.002674 0.004408 0.006485 0.008349 0.009073 0.008957 0.009073 0.008349 0.006485 0.004408 0.002674
0.001812 0.002988 0.004408 0.005814 0.006844 0.007216 0.006844 0.005814 0.004408 0.002988 0.001812
0.001099 0.001812 0.002674 0.003530 0.004170 0.004408 0.004170 0.003530 0.002674 0.001812 0.001099
```

threshold: 0.1

將 input img 與此 kernel 做 cross-correlation，得到的值若大於 threshold 則 output 給值 0，反之給值 255

Result:

Laplace mask1: 20

Laplace mask2: 20

min-variance: 20

LoG: 30

DoG:0.1