電機所-陳治言-R06921084

Description：

一開始，先處理圖片檔的標頭資訊，包括 size，offset，width，height，bits 等資訊，之後再將圖片的色彩值讀入，因為這次範例圖片只有 8 位元色彩，所以每個畫素只需要 1byte 即可儲存。而這次作業是對圖片做邊緣偵測，並使用不同的演算法，處理完圖片之後，再將標頭與畫素資料一起寫出並存，並且產生五張圖片，即可完成這次的作業。

Algorithm：

Laplace Mask1 使用下列的 mask，對圖片進行 convolution。

0	1	0
1	-4	1
0	1	0

Laplace Mask2 使用下列的 mask，對圖片進行 convolution。

$\frac{1}{3}$

1	1	1
1	-8	1
1	1	1

Minimum Variance Laplacian 使用下列的 mask，對圖片進行 convolution。

$\frac{1}{3}$

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

Laplacian of Gaussian 使用下列的 mask，對圖片進行 convolution。

0	0	0	-1	-1	-2	-1	-1	0	0	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	-2	-7	-15	-22	-23	-22	-15	-7	-2	0
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-2	-9	-23	-1	103	178	103	-1	-23	-9	-2
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
0	-2	-7	-15	-22	-23	-22	-15	-7	-2	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	0	0	-1	-1	-2	-1	-1	0	0	0

Difference of Gaussian 使用下列的算式找出相對應的 mask，再對圖片進行 convolution。

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

PrincipalCode：

◎Laplace1

```
for(i=1; i<=bmpInfo.biHeight; ++i)
    for(j=1; j<=bmpInfo.biWidth; ++j)
        p = tmp[i-1][j] + tmp[i+1][j] + tmp[i][j-1] + tmp[i][j+1] - 4 * tmp[i][j];
        BMPoutput_data[i-1][j-1].color = (p > threshold) ? 0 : 255;
```

◎Laplace2

```
for(i=1; i<=bmpInfo.biHeight; ++i)
    for(j=1; j<=bmpInfo.biWidth; ++j)
        p = tmp[i-1][j-1] + tmp[i-1][j] + tmp[i-1][j+1] + tmp[i][j-1] - 8 * tmp[i][j] +
        tmp[i][j+1] + tmp[i+1][j-1] + tmp[i+1][j] + tmp[i+1][j+1];
        BMPoutput_data[i-1][j-1].color = (p / 3 > threshold) ? 0 : 255;
```

◎Minimum_Variance_Laplacian

```
for(i=1; i<=bmpInfo.biHeight; ++i)
    for(j=1; j<=bmpInfo.biWidth; ++j)
        p = 2 * tmp[i-1][j-1] - tmp[i-1][j] + 2 * tmp[i-1][j+1] - tmp[i][j-1] - 4 * tmp[i][j] -
        tmp[i][j+1] + 2 * tmp[i+1][j-1] - tmp[i+1][j] + 2 * tmp[i+1][j+1];
        BMPoutput_data[i-1][j-1].color = (p / 3 > threshold) ? 0 : 255;
```

◎Laplacian_Gaussian

```
//產生 Mask
for(i=0-border; i<=border; ++i)
    for(j=0-border; j<=border; ++j)
        double tmp1 = -175 * (((double)(i * i) + (j * j) - 2 * sigma * sigma) * exp(((double)(i *
        i) + (j * j)) / (-2 * sigma * sigma)) / (sigma * sigma * sigma * sigma));
        LoG[i+border][j+border] = tmp1 > 0 ? tmp1 + 0.5 : tmp1 - 0.5;
--LoG[border][border];

//採用 Zero Padding
for(i=0; i<bmpInfo.biHeight; ++i)
    for(j=0; j<bmpInfo.biWidth; ++j)
        tmp[i+border][j+border] = BMPdata[i][j].color;

for(i=border; i<bmpInfo.biHeight+border; ++i)
    for(j=border; j<bmpInfo.biWidth+border; ++j)
        total = 0;
        for(k=0-border; k<=border; ++k)
            for(l=0-border; l<=border; ++l)
                total += tmp[i+k][j+l] * LoG[border+k][border+l];
        BMPoutput_data[i-border][j-border].color = (total > threshold) ? 0 : 255;
```

◎Difference_Gaussian

```
//產生 Mask
for(i=0-border; i<=border; ++i)
    for(j=0-border; j<=border; ++j)
        DoG[i+border][j+border] = (2000.0 * (Gaussian(i, j, inhibitory_sigma) - Gaussian(i, j,
        excitatory_sigma))) + 0.5;

//採用 Zero Padding
for(i=0; i<bmpInfo.biHeight; ++i)
    for(j=0; j<bmpInfo.biWidth; ++j)
```

```
            tmp[i+border][j+border] = BMPdata[i][j].color;


    for(i=border; i<bmpInfo.biHeight+border; ++i)
        for(j=border; j<bmpInfo.biWidth+border; ++j)
            total = 0;
            for(k=0-border; k<=border; ++k)
                for(l=0-border; l<=border; ++l)
                    total += DoG[border+k][border+l] * tmp[i+k][j+l];
            BMPoutput_data[i-border][j-border].color = (total > threshold) ? 255 : 0;
```

Parameters：

編譯程式碼 g++ -o lena lena.cpp

執行程式./lena lena.bmp

lena.bmp 是我們的 Input Image

ResultingImages：

laplace1_15

laplace2_15

minimum_variance_laplacian_20

laplacian_gaussian_6000

difference_gaussian_20000

$inhibitory\ \sigma = 1 \cdot excitatory\ \sigma = 3 \cdot kernel\ size\ 11 \times 11$