2021/12/26 r10922171 陳嘉政 HW10

- ## Laplace mask 1 (threshold:15)

1. 主要程式碼(每個 mask 步驟都一樣，都會先做個別的
2. mask 函式，再做 zerodetector 函式得出結果圖)

主程式:將 lena 做 padding(與之前作業相同方法)，之後與閥值大小憶起當作參數傳給 Laplacin_mask_1 函式。再將結果傳給 zerodetector 函式，得到最後結果。

```
Mat Laplacin_1 = Laplacin_mask_1(padding(image), 15);
Mat output1 = zerodetector(Laplacin_1);
imshow("output1", output1);
waitKey(0);
imwrite("Laplacin_1.jpg", output1);
```

Laplacin_mask_1 函式:

(1) 宣告一個 **512x512**、全 **0**、但數值形式為有號數的 **output** 矩陣。

(2) 宣告 **Laplace mask**。

(3) 作法與作業 **9** 相同，從第 **1** 列到第 **512** 列，第 **1** 行到第 **512** 行，每個點為中心點畫出 **3x3** 的矩形，與 **mask** 一起傳給 **matrixSum** 函式，回傳值為此矩形的值，若 **>=** 閥值，將 **output** 中對應點**(i-1,j-1)**設為 **1**；若 **<=** 負閥值，設為 **-1**；否則設為 **0**。最後回傳 **output** 矩陣。

```
Mat Laplacin_mask_1(Mat padded, int threshold)
{
    Mat output = Mat::zeros(512, 512, CV_8SC1);

    Mat mask = (Mat_<double>(3, 3) <<
        0,  1, 0,
        1, -4, 1,
        0,  1, 0);

    for (int i = 1; i < padded.rows - 1; i++) {
        for (int j = 1; j < padded.cols - 1; j++) {
            Mat kernel = padded(cv::Rect(j - 1, i - 1, 3, 3));
            double value = matrixSum(kernel, mask);
            if (value >= threshold)
                output.at<int8_t>(i - 1, j - 1) = 1;
            else if (value <= -threshold)
                output.at<int8_t>(i - 1, j - 1) = -1;
            else
                output.at<int8_t>(i - 1, j - 1) = 0;
        }
    }
    return output;
}
```

matrixSum 函式:

與作業 **9** 相同。用來計算兩矩陣對應點相乘後加總的值。

```
double matrixSum(Mat kernel, Mat mask)
{
    double total = 0;
    for (int i = 0; i < kernel.rows; i++) {
        for (int j = 0; j < kernel.cols; j++) {
            total += kernel.at<uint8_t>(i, j) * mask.at<double>(i, j);
        }
    }
    return total;
}
```

Zerodetector 函式:

(1) 將輸入做有號數版本的 padding(因為先前的輸出式有號數矩陣，有可能有-1)，設給 padded。

(2) 宣告一個 512x512、pixel 值皆為 255 的 result matrix。

(3) Zerodetector 只看該點周圍 8 個鄰居，所以我們從第 1 列到 512 列，第 1 行到第 512 行，每個點檢查，若該點的 value 為 1，表示要接著檢查鄰居，若該點的鄰居有任一個點 value 為 -1 時，表示 value 相差很大，該點應該被當成邊，就將該 result 中點(I-1, j-1)設為 0；若此點非 1 或是此點鄰居皆 > -1，則 result 中點(I-1, j-1)設為 255。最後回傳 result matrix 即為所求結果。

```
Mat zerodetector(Mat input)
{
    Mat padded = signed_padding(input);

    Mat result = Mat(512, 512, CV_8UC1, Scalar(255));
    for (int i = 1; i < padded.rows - 1; i++) {
        for (int j = 1; j < padded.cols - 1; j++) {
            if (padded.at<int8_t>(i, j) == 1) {
                for (int dr = -1; dr <= 1; dr++) {
                    for (int dc = -1; dc <= 1; dc++) {
                        if (dr != 0 || dc != 0) {
                            if (padded.at<int8_t>(i + dr, j + dc) == -1) {
                                result.at<uint8_t>(i - 1, j - 1) = 0;
                                break;
                            }
                            else
                                result.at<uint8_t>(i - 1, j - 1) = 255;
                        }
                    }
                    if (result.at<uint8_t>(i - 1, j - 1) == 0)
                        break;
                }
            }
            else
                result.at<uint8_t>(i - 1, j - 1) = 255;
        }
    }
    return result;
}
```

3. 結果

- **Laplace mask 2 (threshold:15)**

1. 主要程式碼(只貼不同的函式及程式碼)

主程式:

```
Mat Laplacin_2 = Laplacin_mask_2(padding(image), 15);
Mat output2 = zerodetector(Laplacin_2);
imshow("output2", output2);
waitKey(0);
imwrite("Laplacin_2.jpg", output2);
```

Laplacin_mask_2:

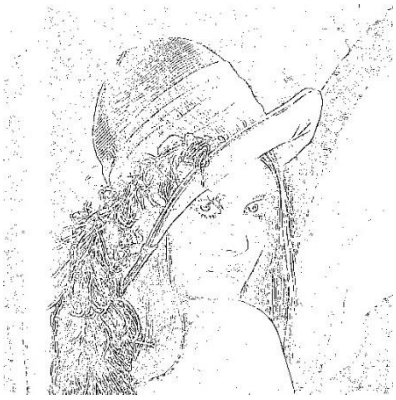
差別在於 **mask** 數值不同。

```
Mat Laplacin_mask_2(Mat padded, int threshold)
{
    Mat output = Mat::zeros(512, 512, CV_8SC1);

    Mat mask = ((double)1.0/3) * (Mat_<double>(3, 3) <<
        1,  1, 1,
        1, -8, 1,
        1,  1, 1);

    for (int i = 1; i < padded.rows - 1; i++) {
        for (int j = 1; j < padded.cols - 1; j++) {
            Mat kernel = padded(cv::Rect(j - 1, i - 1, 3, 3));
            double value = matrixSum(kernel, mask);
            if (value >= threshold)
                output.at<int8_t>(i - 1, j - 1) = 1;
            else if (value <= -threshold)
                output.at<int8_t>(i - 1, j - 1) = -1;
            else
                output.at<int8_t>(i - 1, j - 1) = 0;
        }
    }
    return output;
}
```

2. 結果



- **Minimum variance Laplacin (threshold:20)**

1. 主要程式碼(只貼不同的函式及程式碼)

主程式:

```
Mat minimum_variance_Laplacin = minimum_variance_Laplacin_mask(padding(image), 20);
Mat output3 = zerodetector(minimum_variance_Laplacin);
imshow("output3", output3);
waitKey(0);
imwrite("minimum_variance_Laplacin.jpg", output3);
```

Minimum_variance_Laplacin_mask:

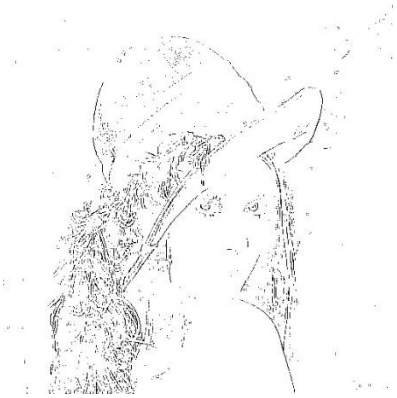
差別在於 **mask** 數值不同。

```
Mat minimum_variance_Laplacin_mask(Mat padded, int threshold)
{
    Mat output = Mat::zeros(512, 512, CV_8SC1);

    Mat mask = 1.0 / 3 * (Mat_<double>(3, 3) <<
         2, -1,  2,
        -1, -4, -1,
         2, -1,  2);

    for (int i = 1; i < padded.rows - 1; i++) {
        for (int j = 1; j < padded.cols - 1; j++) {
            Mat kernel = padded(cv::Rect(j - 1, i - 1, 3, 3));
            double value = matrixSum(kernel, mask);
            if (value >= threshold)
                output.at<int8_t>(i - 1, j - 1) = 1;
            else if (value <= -threshold)
                output.at<int8_t>(i - 1, j - 1) = -1;
            else
                output.at<int8_t>(i - 1, j - 1) = 0;
        }
    }
    return output;
}
```

2. 結果



- **Laplacian of Gaussian (threshold:3000)**

1. 主要程式碼(只貼不同的函式及程式碼)

   主程式:

   原 **lena** 的 **padding** 要做 **5** 次。

```
Mat Laplacin_of_Gaussian = Laplacin_of_Gaussian_mask(padding(padding(padding(padding(padding(image))))), 3000);
Mat output4 = zerodetector(Laplacin_of_Gaussian);
imshow("output4", output4);
waitKey(0);
imwrite("Laplacin_of_Gaussian.jpg", output4);
```

   Laplacin_of_Gaussian_mask:

   (1) mask 大小**(11x11)**及數值不同。

   (2) for loop 範圍也跟著改變。

   (3) 圈出的矩形為 **11x11**。

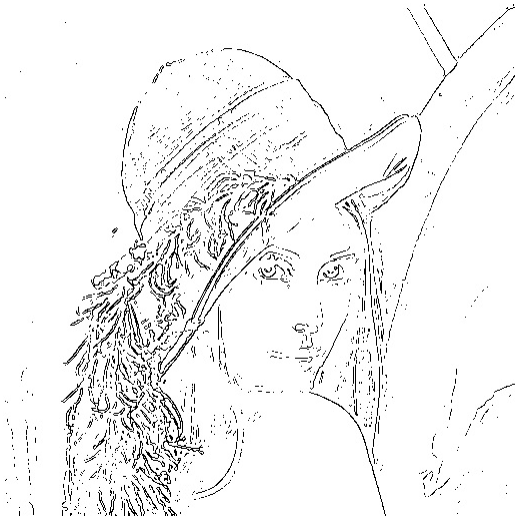
   (4) Output 對應的點為**(I – 5, j – 5)**。

```
Mat Laplacin_of_Gaussian_mask(Mat padded, int threshold)
{
    Mat output = Mat::zeros(512, 512, CV_8SC1);

    Mat mask = (Mat_<double>(11, 11) <<
         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);

    for (int i = 5; i < padded.rows - 5; i++) {
        for (int j = 5; j < padded.cols - 5; j++) {
            Mat kernel = padded(cv::Rect(j - 5, i - 5, 11, 11));
            double value = matrixSum(kernel, mask);
            if (value >= threshold)
                output.at<int8_t>(i - 5, j - 5) = 1;
            else if (value <= -threshold)
                output.at<int8_t>(i - 5, j - 5) = -1;
            else
                output.at<int8_t>(i - 5, j - 5) = 0;
        }
    }
    return output;
}
```

2. 結果



- **Difference of Gaussian (threshold:1)**

1. 主要程式碼(只貼不同的函式及程式碼)

   主程式:

   與前個相同，**lena** 要 **padding 5** 次。

```
Mat difference_of_Gaussian = difference_of_Gaussian_mask(padding(padding(padding(padding(padding(image))))), 1);
Mat output5 = zerodetector(difference_of_Gaussian);
imshow("output5", output5);
waitKey(0);
imwrite("difference_of_Gaussian.jpg", output5);
```

   Difference_of_Gaussian_mask:

   只有 **mask** 的值與前一個不同，其他都和前一個相同。

```
Mat difference_of_Gaussian_mask(Mat padded, int threshold)
{
    Mat output = Mat::zeros(512, 512, CV_8SC1);

    Mat mask = (Mat_<double>(11, 11) <<
        -1,  -3,  -4, -6,  -7,  -8,  -7, -6,  -4,  -3, -1,
        -3,  -5,  -8,-11, -13, -13, -13,-11,  -8,  -5, -3,
        -4,  -8, -12,-16, -17, -17, -17,-16, -12,  -8, -4,
        -6, -11, -16,-16,   0,  15,   0,-16, -16, -11, -6,
        -7, -13, -17,  0,  85, 160,  85,  0, -17, -13, -7,
        -8, -13, -17, 15, 160, 283, 160, 15, -17, -13, -8,
        -7, -13, -17,  0,  85, 160,  85,  0, -17, -13, -7,
        -6, -11, -16,-16,   0,  15,   0,-16, -16, -11, -6,
        -4,  -8, -12,-16, -17, -17, -17,-16, -12,  -8, -4,
        -3,  -5,  -8,-11, -13, -13, -13,-11,  -8,  -5, -3,
        -1,  -3,  -4, -6,  -7,  -8,  -7, -6,  -4,  -3, -1);

    for (int i = 5; i < padded.rows - 5; i++) {
        for (int j = 5; j < padded.cols - 5; j++) {
            Mat kernel = padded(cv::Rect(j - 5, i - 5, 11, 11));
            double value = matrixSum(kernel, mask);
            if (value >= threshold)
                output.at<int8_t>(i - 5, j - 5) = 1;
            else if (value <= -threshold)
                output.at<int8_t>(i - 5, j - 5) = -1;
            else
                output.at<int8_t>(i - 5, j - 5) = 0;
        }
    }
    return output;
```

2. 結果