Homework 9

General Edge Detection

R09921119 許凱荃

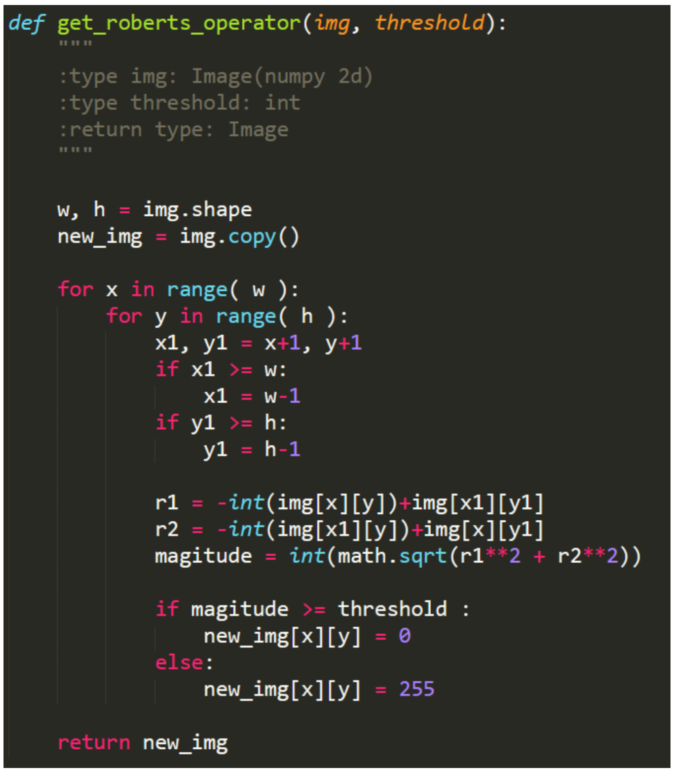
- You are to implement following edge detectors with thresholds :
  - (a) Robert's Operator: 12
  - (b) Prewitt's Edge Detector: 24
  - (c) Sobel's Edge Detector: 38
  - (d) Frei and Chen's Gradient Operator: 30
  - (e) Kirsch's Compass Operator: 135
  - (f) Robinson's Compass Operator: 43
  - (g) Nevatia-Babu 5x5 Operator: 12500

Outline :

- Method
- Image Result

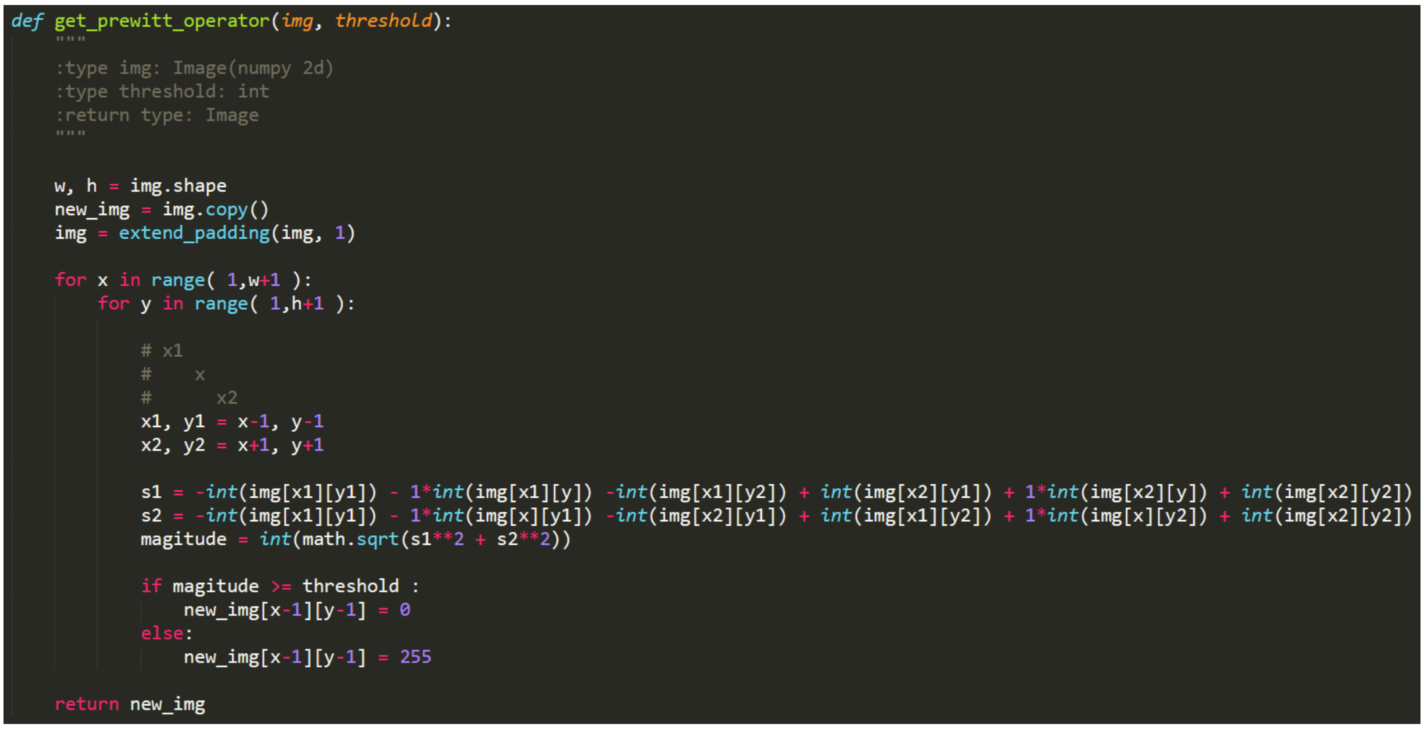
Method

- **Robert's Operator**：計算矩陣，和其他比較不同的是它是 2 * 2，沒有一個 Center；故不需要 extend padding，若有超出邊界的座標直接縮回來。



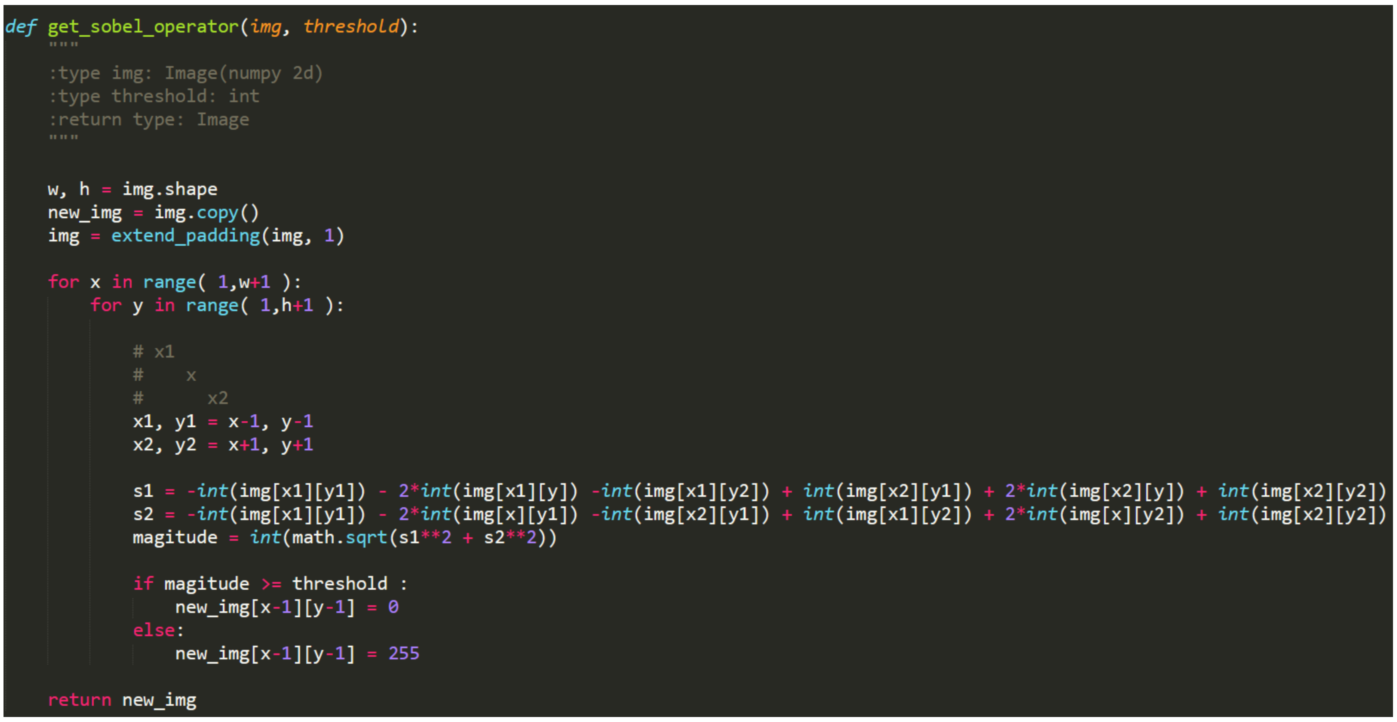
```python
def get_roberts_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()

    for x in range( w ):
        for y in range( h ):
            x1, y1 = x+1, y+1
            if x1 >= w:
                x1 = w-1
            if y1 >= h:
                y1 = h-1

            r1 = -int(img[x][y])+img[x1][y1]
            r2 = -int(img[x1][y])+img[x][y1]
            magitude = int(math.sqrt(r1**2 + r2**2))

            if magitude >= threshold :
                new_img[x][y] = 0
            else:
                new_img[x][y] = 255

    return new_img
```

- **Prewitt's Edge Detector**：按照 table 計算權重。



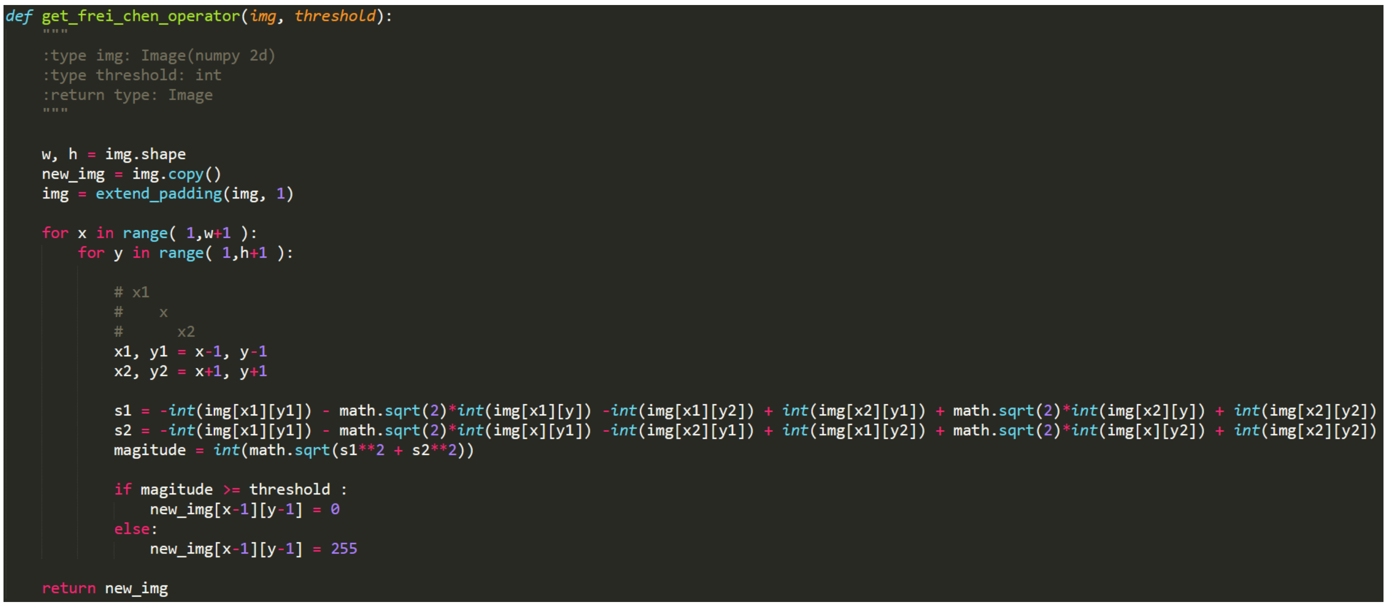
```python
def get_prewitt_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1
            #    x
            #      x2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            s1 = -int(img[x1][y1]) - 1*int(img[x1][y]) -int(img[x1][y2]) + int(img[x2][y1]) + 1*int(img[x2][y]) + int(img[x2][y2])
            s2 = -int(img[x1][y1]) - 1*int(img[x][y1]) -int(img[x2][y1]) + int(img[x1][y2]) + 1*int(img[x][y2]) + int(img[x2][y2])
            magitude = int(math.sqrt(s1**2 + s2**2))

            if magitude >= threshold :
                new_img[x-1][y-1] = 0
            else:
                new_img[x-1][y-1] = 255

    return new_img
```

- **Sobel's Edge Detector**：按照 table 計算權重。



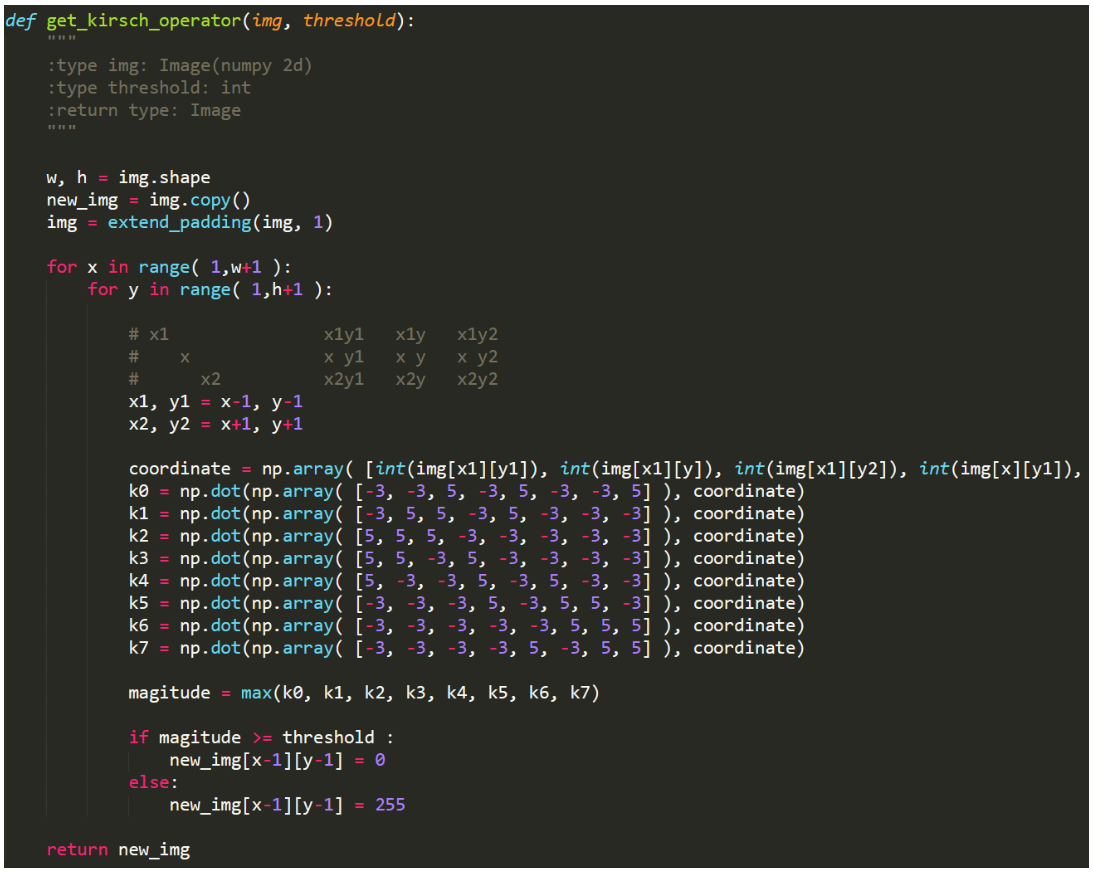
```python
def get_sobel_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1
            #    x
            #       x2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            s1 = -int(img[x1][y1]) - 2*int(img[x1][y]) -int(img[x1][y2]) + int(img[x2][y1]) + 2*int(img[x2][y]) + int(img[x2][y2])
            s2 = -int(img[x1][y1]) - 2*int(img[x][y1]) -int(img[x2][y1]) + int(img[x1][y2]) + 2*int(img[x][y2]) + int(img[x2][y2])
            magitude = int(math.sqrt(s1**2 + s2**2))

            if magitude >= threshold :
                new_img[x-1][y-1] = 0
            else:
                new_img[x-1][y-1] = 255

    return new_img
```

- **Frei and Chen's Gradient Operator**：按照 table 計算權重。



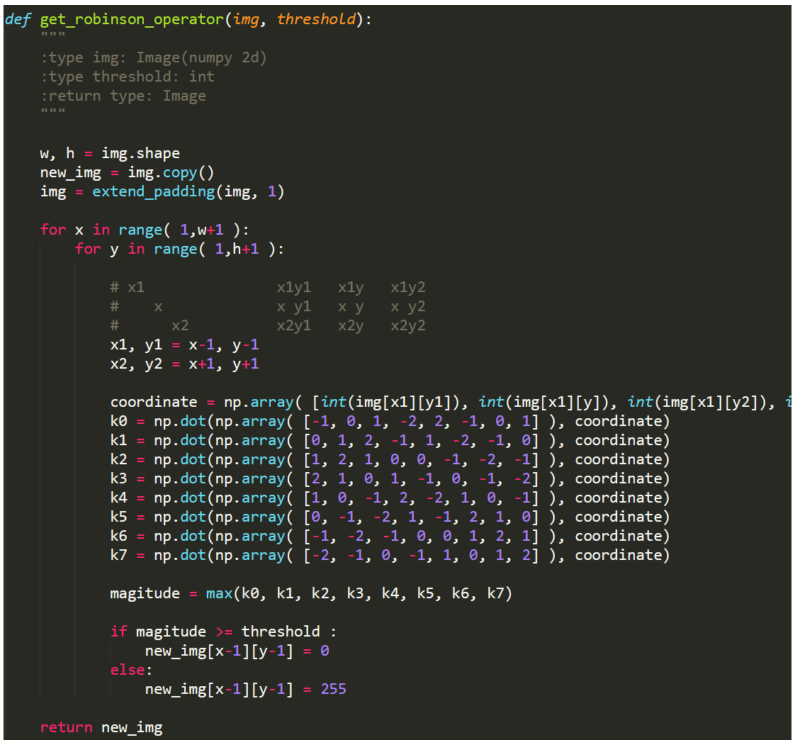
```python
def get_frei_chen_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1
            #    x
            #       x2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            s1 = -int(img[x1][y1]) - math.sqrt(2)*int(img[x1][y]) -int(img[x1][y2]) + int(img[x2][y1]) + math.sqrt(2)*int(img[x2][y]) + int(img[x2][y2])
            s2 = -int(img[x1][y1]) - math.sqrt(2)*int(img[x][y1]) -int(img[x2][y1]) + int(img[x1][y2]) + math.sqrt(2)*int(img[x][y2]) + int(img[x2][y2])
            magitude = int(math.sqrt(s1**2 + s2**2))

            if magitude >= threshold :
                new_img[x-1][y-1] = 0
            else:
                new_img[x-1][y-1] = 255

    return new_img
```

- **Kirsch's Compass Operator**：
  一樣是照 Table 填，但是改了方法，將九宮格座標之值弄成陣列，與係數陣列做內積。這樣做會讓排版好看很多，debug 也比較好找，美觀 ！



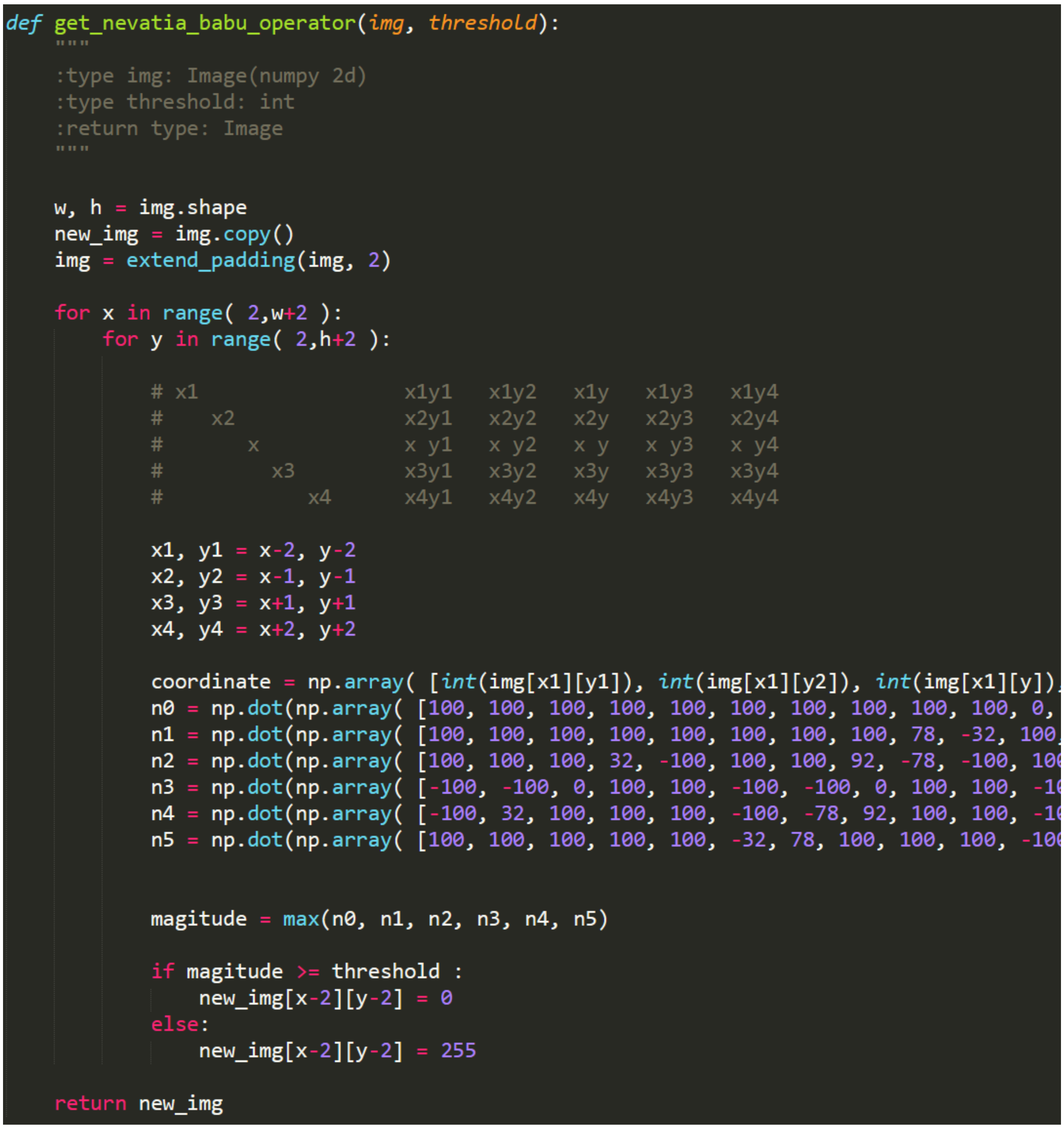
```python
def get_kirsch_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1                x1y1   x1y   x1y2
            #    x              x y1   x y   x y2
            #      x2           x2y1   x2y   x2y2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y]), int(img[x1][y2]), int(img[x][y1]),
            k0 = np.dot(np.array( [-3, -3, 5, -3, 5, -3, -3, 5] ), coordinate)
            k1 = np.dot(np.array( [-3, 5, 5, -3, 5, -3, -3, -3] ), coordinate)
            k2 = np.dot(np.array( [5, 5, 5, -3, -3, -3, -3, -3] ), coordinate)
            k3 = np.dot(np.array( [5, 5, -3, 5, -3, -3, -3, -3] ), coordinate)
            k4 = np.dot(np.array( [5, -3, -3, 5, -3, 5, -3, -3] ), coordinate)
            k5 = np.dot(np.array( [-3, -3, -3, 5, -3, 5, 5, -3] ), coordinate)
            k6 = np.dot(np.array( [-3, -3, -3, -3, -3, 5, 5, 5] ), coordinate)
            k7 = np.dot(np.array( [-3, -3, -3, -3, 5, -3, 5, 5] ), coordinate)

            magitude = max(k0, k1, k2, k3, k4, k5, k6, k7)

            if magitude >= threshold :
                new_img[x-1][y-1] = 0
            else:
                new_img[x-1][y-1] = 255

    return new_img
```

- **Robinson's Compass Operator**：按照 table 計算權重。



```python
def get_robinson_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    img = extend_padding(img, 1)

    for x in range( 1,w+1 ):
        for y in range( 1,h+1 ):

            # x1                x1y1   x1y   x1y2
            #    x              x y1   x y   x y2
            #      x2           x2y1   x2y   x2y2
            x1, y1 = x-1, y-1
            x2, y2 = x+1, y+1

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y]), int(img[x1][y2]), i
            k0 = np.dot(np.array( [-1, 0, 1, -2, 2, -1, 0, 1] ), coordinate)
            k1 = np.dot(np.array( [0, 1, 2, -1, 1, -2, -1, 0] ), coordinate)
            k2 = np.dot(np.array( [1, 2, 1, 0, 0, -1, -2, -1] ), coordinate)
            k3 = np.dot(np.array( [2, 1, 0, 1, -1, 0, -1, -2] ), coordinate)
            k4 = np.dot(np.array( [1, 0, -1, 2, -2, 1, 0, -1] ), coordinate)
            k5 = np.dot(np.array( [0, -1, -2, 1, -1, 2, 1, 0] ), coordinate)
            k6 = np.dot(np.array( [-1, -2, -1, 0, 0, 1, 2, 1] ), coordinate)
            k7 = np.dot(np.array( [-2, -1, 0, -1, 1, 0, 1, 2] ), coordinate)

            magitude = max(k0, k1, k2, k3, k4, k5, k6, k7)

            if magitude >= threshold :
                new_img[x-1][y-1] = 0
            else:
                new_img[x-1][y-1] = 255

    return new_img
```

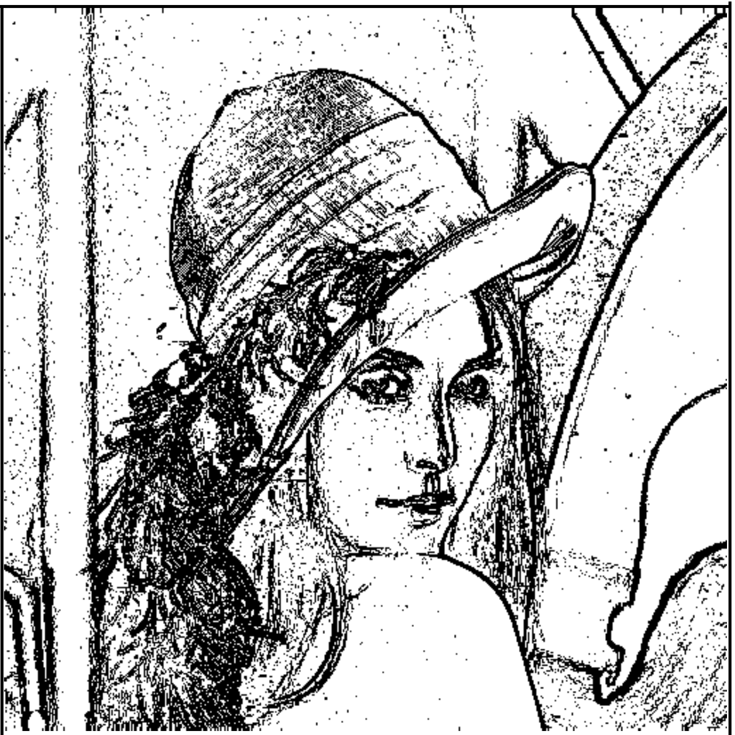
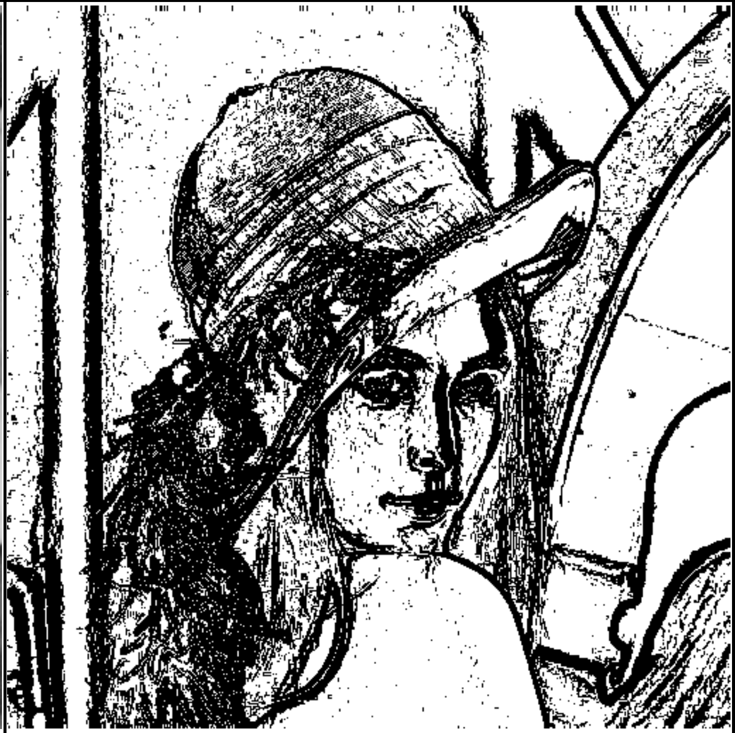
- **Nevatia-Babu 5x5 Operator**：按照 table 計算權重。
  五乘五矩陣很長一坨 ...。

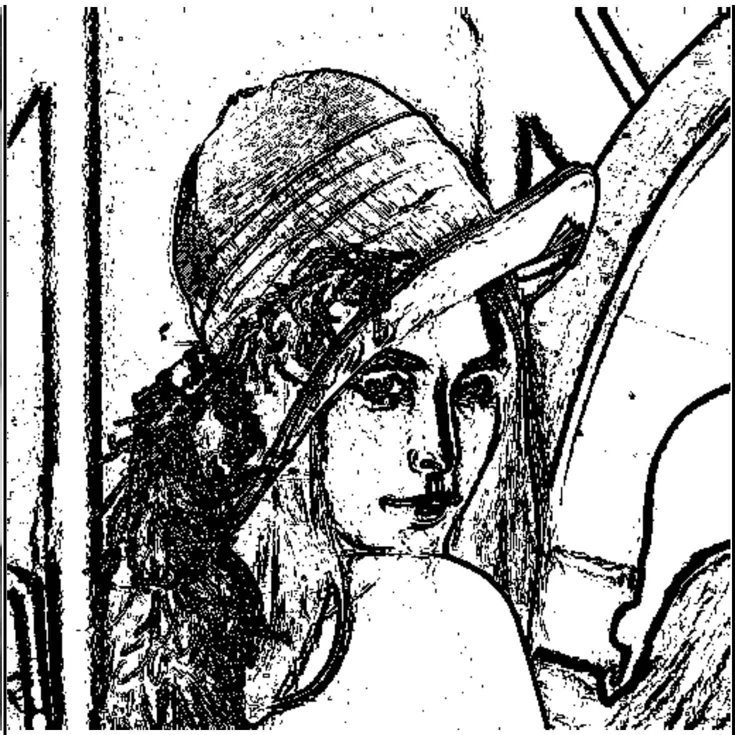
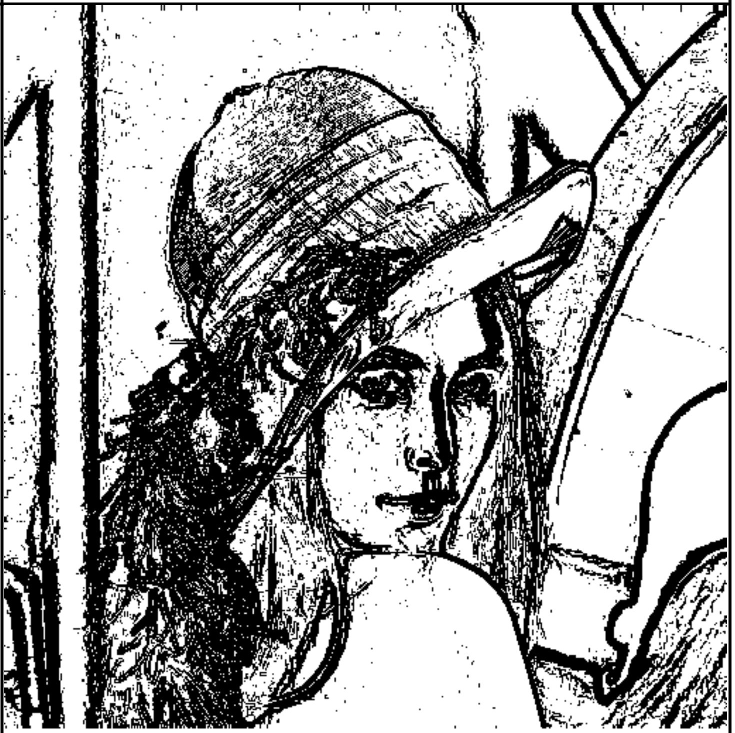


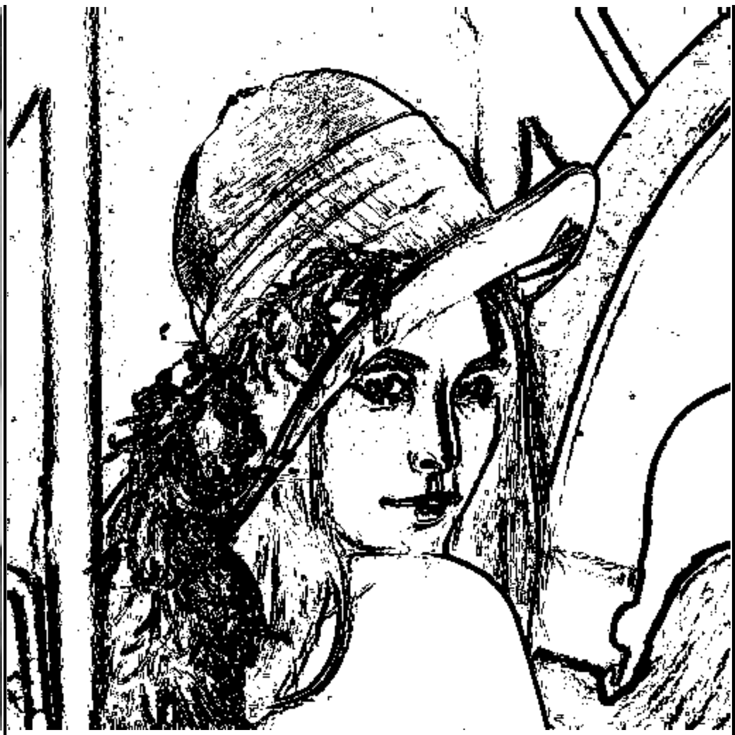
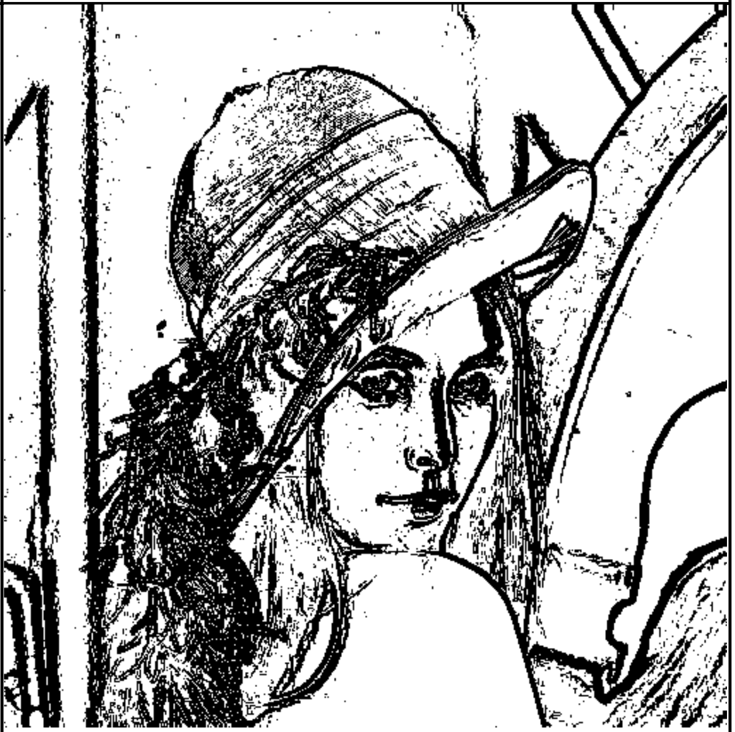
```
def get_nevatia_babu_operator(img, threshold):
    """
    :type img: Image(numpy 2d)
    :type threshold: int
    :return type: Image
    """

    w, h = img.shape
    new_img = img.copy()
    img = extend_padding(img, 2)

    for x in range( 2,w+2 ):
        for y in range( 2,h+2 ):

            # x1                    x1y1   x1y2   x1y   x1y3   x1y4
            #    x2                 x2y1   x2y2   x2y   x2y3   x2y4
            #       x               x y1   x y2   x y   x y3   x y4
            #         x3            x3y1   x3y2   x3y   x3y3   x3y4
            #            x4         x4y1   x4y2   x4y   x4y3   x4y4

            x1, y1 = x-2, y-2
            x2, y2 = x-1, y-1
            x3, y3 = x+1, y+1
            x4, y4 = x+2, y+2

            coordinate = np.array( [int(img[x1][y1]), int(img[x1][y2]), int(img[x1][y]),
            n0 = np.dot(np.array( [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 0,
            n1 = np.dot(np.array( [100, 100, 100, 100, 100, 100, 100, 100, 78, -32, 100
            n2 = np.dot(np.array( [100, 100, 100, 32, -100, 100, 100, 92, -78, -100, 10
            n3 = np.dot(np.array( [-100, -100, 0, 100, 100, -100, -100, 0, 100, 100, -1
            n4 = np.dot(np.array( [-100, 32, 100, 100, 100, -100, -78, 92, 100, 100, -1
            n5 = np.dot(np.array( [100, 100, 100, 100, 100, -32, 78, 100, 100, 100, -10


            magitude = max(n0, n1, n2, n3, n4, n5)

            if magitude >= threshold :
                new_img[x-2][y-2] = 0
            else:
                new_img[x-2][y-2] = 255

    return new_img
```

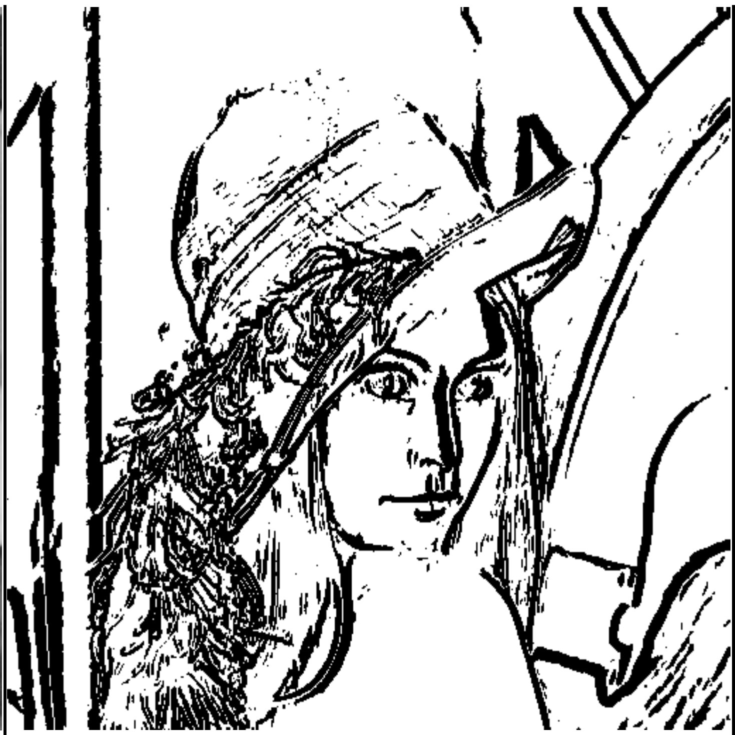
Image Result

- **Gaussian noise, amplitude of 10**

	
Original Lena	Robert's Operator: 12
	
Original Lena	Prewitt's Edge Detector: 24

	
Original Lena	Sobel's Edge Detector: 38
	
Original Lena	Frei and Chen's Gradient Operator: 30

	
Original Lena	Kirsch's Compass Operator: 135
	
Original Lena	Robinson's Compass Operator: 43

	
Original Lena	**Nevatia-Babu 5x5 Operator: 12500**

Tips : 用作業說明 PDF 的矩陣跑跑看當作 Debug，會讓此作業順利許多

助教辛苦了 :)

🍵 (´ω`)