HW9 report

姓名:徐有慶　學號:R05922162

1. 作業要求:

General Edge Detection

- You are to implement Robert, Prewitt, Sobel, Frei & Chen, Kirsch, Robinson, and Nevatia-Babu’s edge detectors.
- Threshold Values listed below are for reference:

  (僅供參考，同學可自己找出 Edge Image 品質最佳的門檻值 threshold value)

  – Robert’s Operator: 12

  – Prewitt’s Edge Detector: 24

  – Sobel’s Edge Detector: 38

  – Frei and Chen’s Gradient Operator: 30

  – Kirsch’s Compass Operator: 135

  – Robinson’s Compass Operator: 43

  – Nevatia-Babu 5x5 Operator: 12500

  程式語言：Java

  執行環境：Eclipse

2. 程式設計:

- Roberts operators:

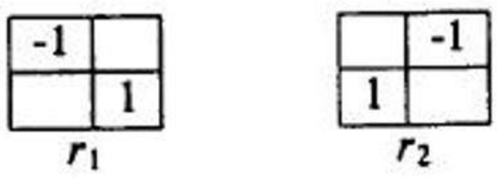
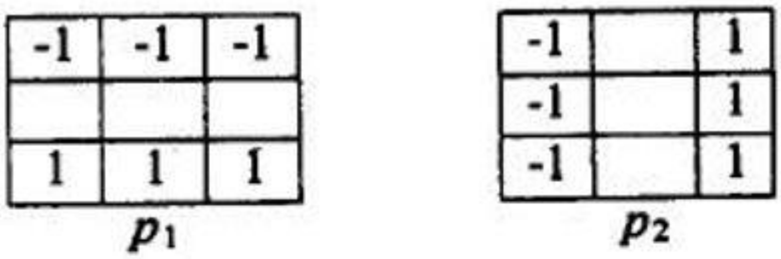


Figure 7.21 Masks used for the Roberts operators.

把 r1、r2 當作 mask，左上角的點當作目前處理的 pixel。計算$(Gr1^2 + Gr2^2)^{1/2}$當作目前 pixel 的值，其中 Gr1，Gr2 分別由 r1 及 r2 得出。

- Prewitt edge detector:



-1	-1	-1
1	1	1

p_1

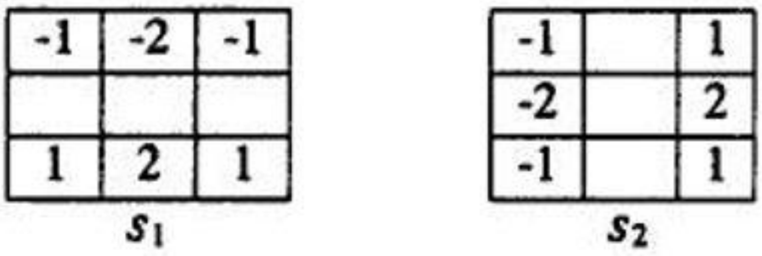
-1		1
-1		1
-1		1

p_2

Figure 7.22 Prewitt edge detector masks.

中心點當作目前處理的 pixel，其餘計算方法同 Roberts operators

- Sobel edge detector:



-1	-2	-1
1	2	1

s_1

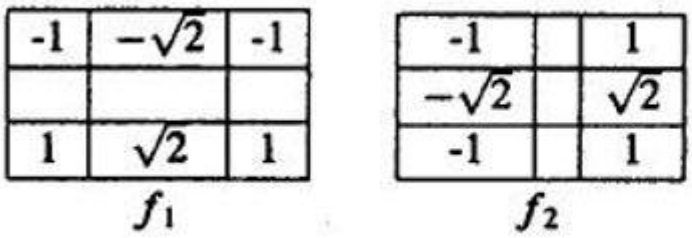
-1		1
-2		2
-1		1

s_2

Figure 7.23 Sobel edge detector masks.

計算方法同 Prewitt edge detector

- Frei and Chen edge dector:



-1	$-\sqrt{2}$	-1
1	$\sqrt{2}$	1

f_1

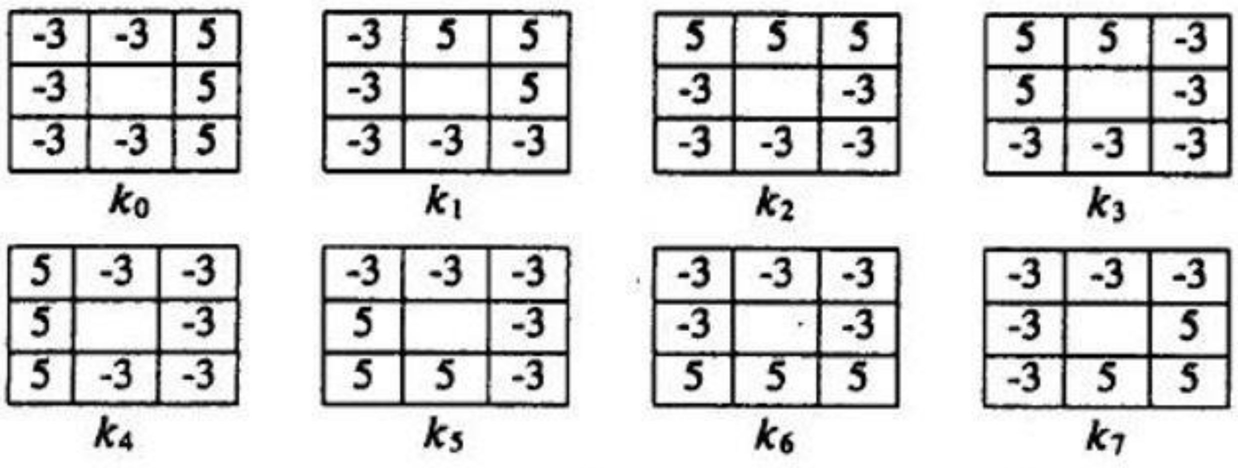
-1		1
$-\sqrt{2}$		$\sqrt{2}$
-1		1

f_2

Figure 7.24 Frei and Chen gradient masks.

計算方法同 Prewitt edge detector

- Kirsch operators:



-3	-3	5
-3		5
-3	-3	5

k_0

-3	5	5
-3		5
-3	-3	-3

k_1

5	5	5
-3		-3
-3	-3	-3

k_2

5	5	-3
5		-3
-3	-3	-3

k_3

5	-3	-3
5		-3
5	-3	-3

k_4

-3	-3	-3
5		-3
5	5	-3

k_5

-3	-3	-3
-3		-3
5	5	5

k_6

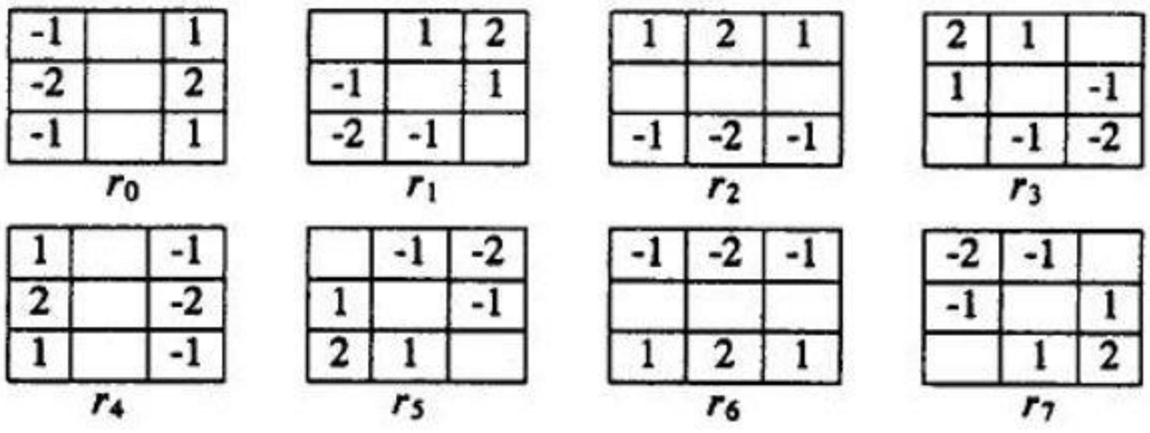
-3	-3	-3
-3		5
-3	5	5

k_7

Figure 7.25 Kirsch compass masks.

gradient magnitude: $g = \max_{n,n=0,\ldots,7} k_n$

- Robinson operators:



-1		1
-2		2
-1		1

r_0

	1	2
-1		1
-2	-1	

r_1

1	2	1
-1	-2	-1

r_2

2	1	
1		-1
	-1	-2

r_3

1		-1
2		-2
1		-1

r_4

	-1	-2
1		-1
2	1	

r_5

-1	-2	-1
1	2	1

r_6

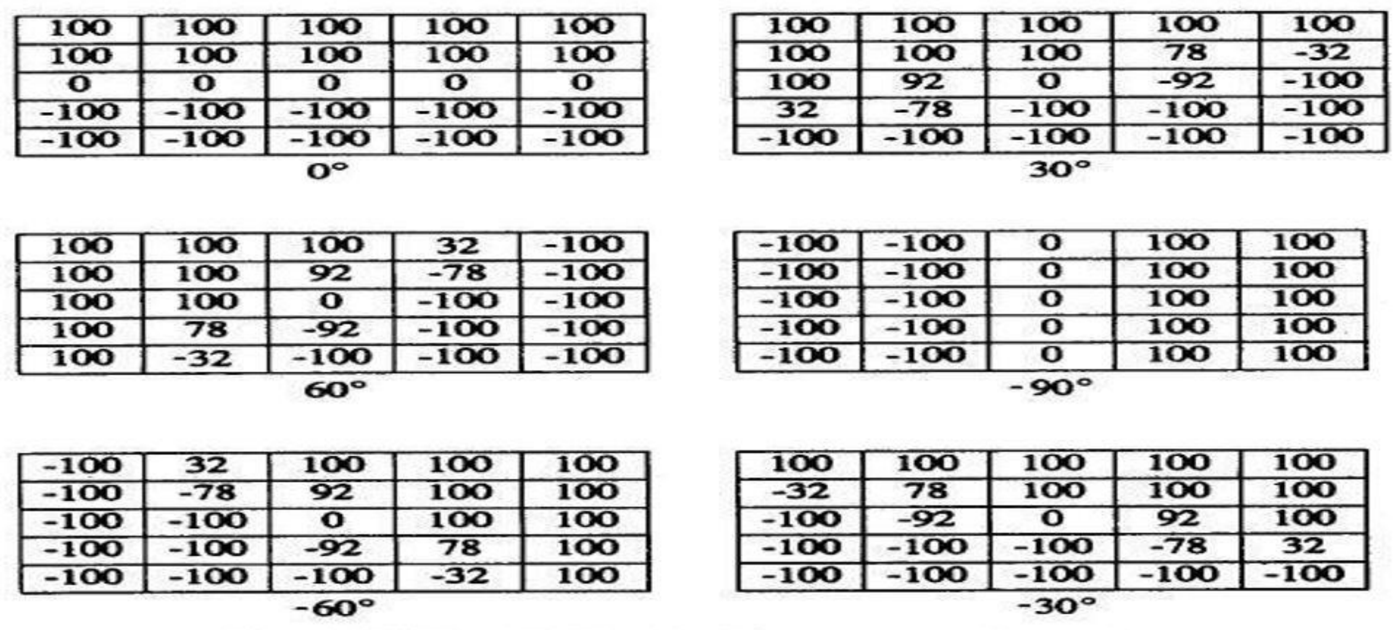
-2	-1	
-1		1
	1	2

r_7

Figure 7.26 Robinson compass masks.

計算方法同 Kirsch operators

- Nevatia and Babu operators:



100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

0°

100	100	100	100	100
100	100	100	78	-32
100	92	0	-92	-100
32	-78	-100	-100	-100
-100	-100	-100	-100	-100

30°

100	100	100	32	-100
100	100	92	-78	-100
100	100	0	-100	-100
100	78	-92	-100	-100
100	-32	-100	-100	-100

60°

-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100

-90°

-100	32	100	100	100
-100	-78	92	100	100
-100	-100	0	100	100
-100	-100	-92	78	100
-100	-100	-100	-32	100

-60°

100	100	100	100	100
-32	78	100	100	100
-100	-92	0	92	100
-100	-100	-100	-78	32
-100	-100	-100	-100	-100

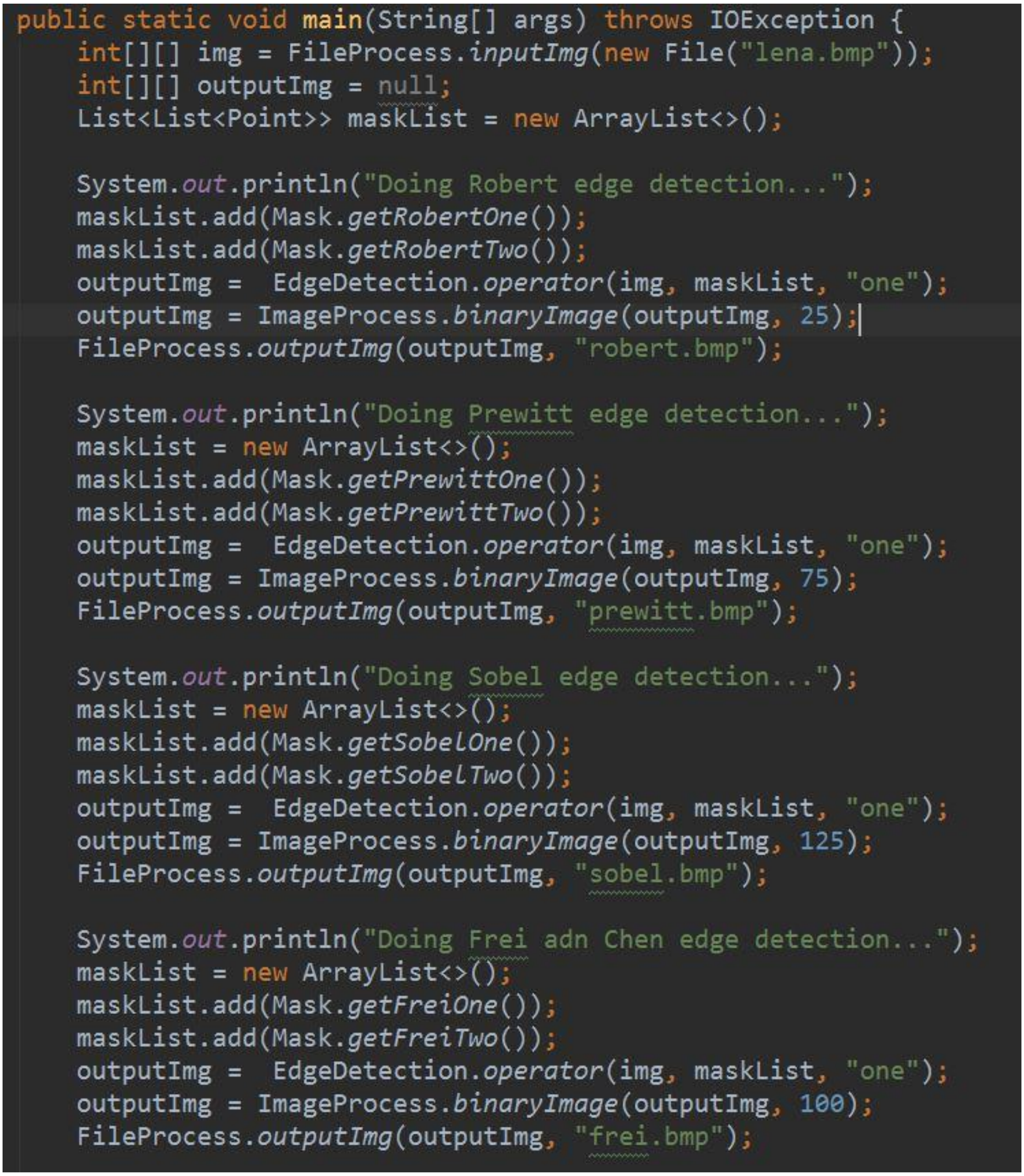
-30°

Figure 7.27 Nevatia-Babu 5 × 5 compass template masks.

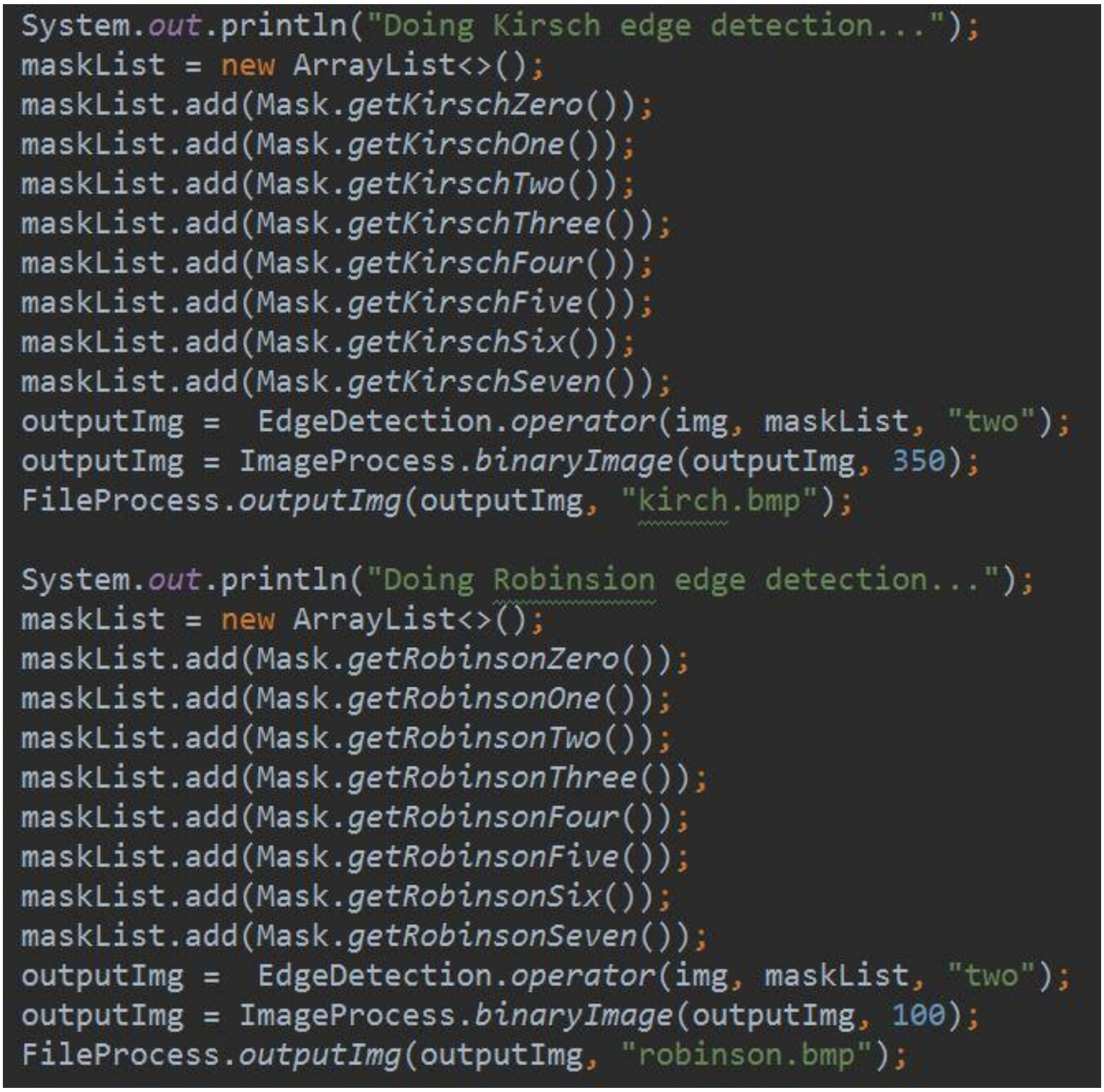
計算方法同 Kirsch operators

3. 主要程式:

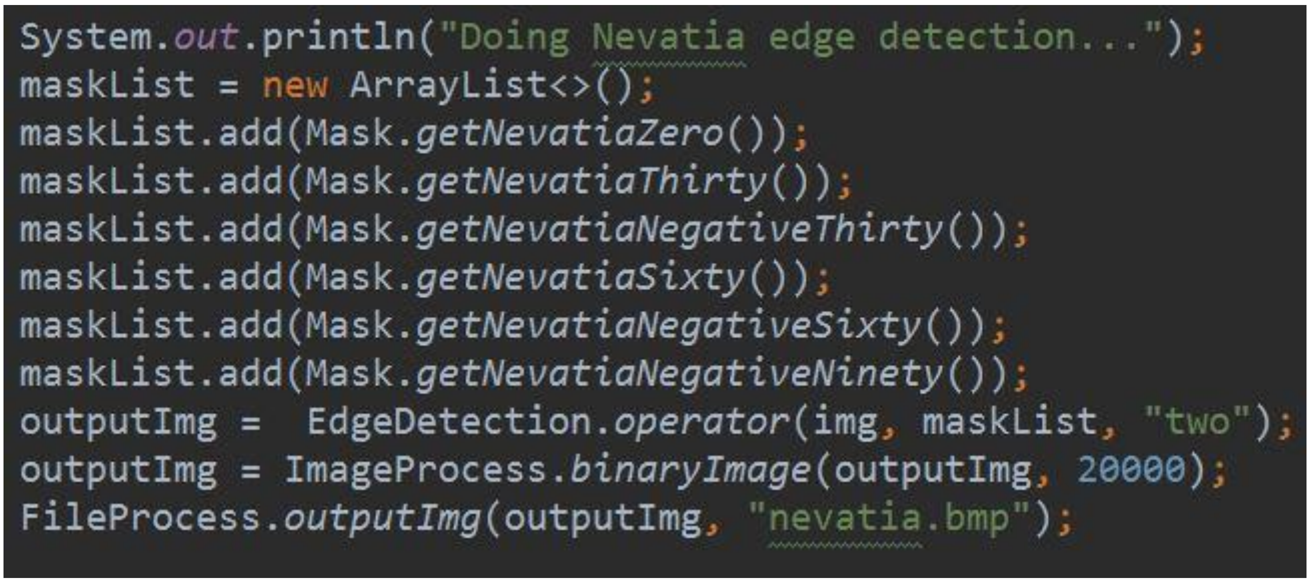
Main(./Source Code/src/hw9/MainEdgeDetection.java):



```java
public static void main(String[] args) throws IOException {
    int[][] img = FileProcess.inputImg(new File("lena.bmp"));
    int[][] outputImg = null;
    List<List<Point>> maskList = new ArrayList<>();

    System.out.println("Doing Robert edge detection...");
    maskList.add(Mask.getRobertOne());
    maskList.add(Mask.getRobertTwo());
    outputImg =  EdgeDetection.operator(img, maskList, "one");
    outputImg = ImageProcess.binaryImage(outputImg, 25);
    FileProcess.outputImg(outputImg, "robert.bmp");

    System.out.println("Doing Prewitt edge detection...");
    maskList = new ArrayList<>();
    maskList.add(Mask.getPrewittOne());
    maskList.add(Mask.getPrewittTwo());
    outputImg =  EdgeDetection.operator(img, maskList, "one");
    outputImg = ImageProcess.binaryImage(outputImg, 75);
    FileProcess.outputImg(outputImg, "prewitt.bmp");

    System.out.println("Doing Sobel edge detection...");
    maskList = new ArrayList<>();
    maskList.add(Mask.getSobelOne());
    maskList.add(Mask.getSobelTwo());
    outputImg =  EdgeDetection.operator(img, maskList, "one");
    outputImg = ImageProcess.binaryImage(outputImg, 125);
    FileProcess.outputImg(outputImg, "sobel.bmp");

    System.out.println("Doing Frei adn Chen edge detection...");
    maskList = new ArrayList<>();
    maskList.add(Mask.getFreiOne());
    maskList.add(Mask.getFreiTwo());
    outputImg =  EdgeDetection.operator(img, maskList, "one");
    outputImg = ImageProcess.binaryImage(outputImg, 100);
    FileProcess.outputImg(outputImg, "frei.bmp");
```

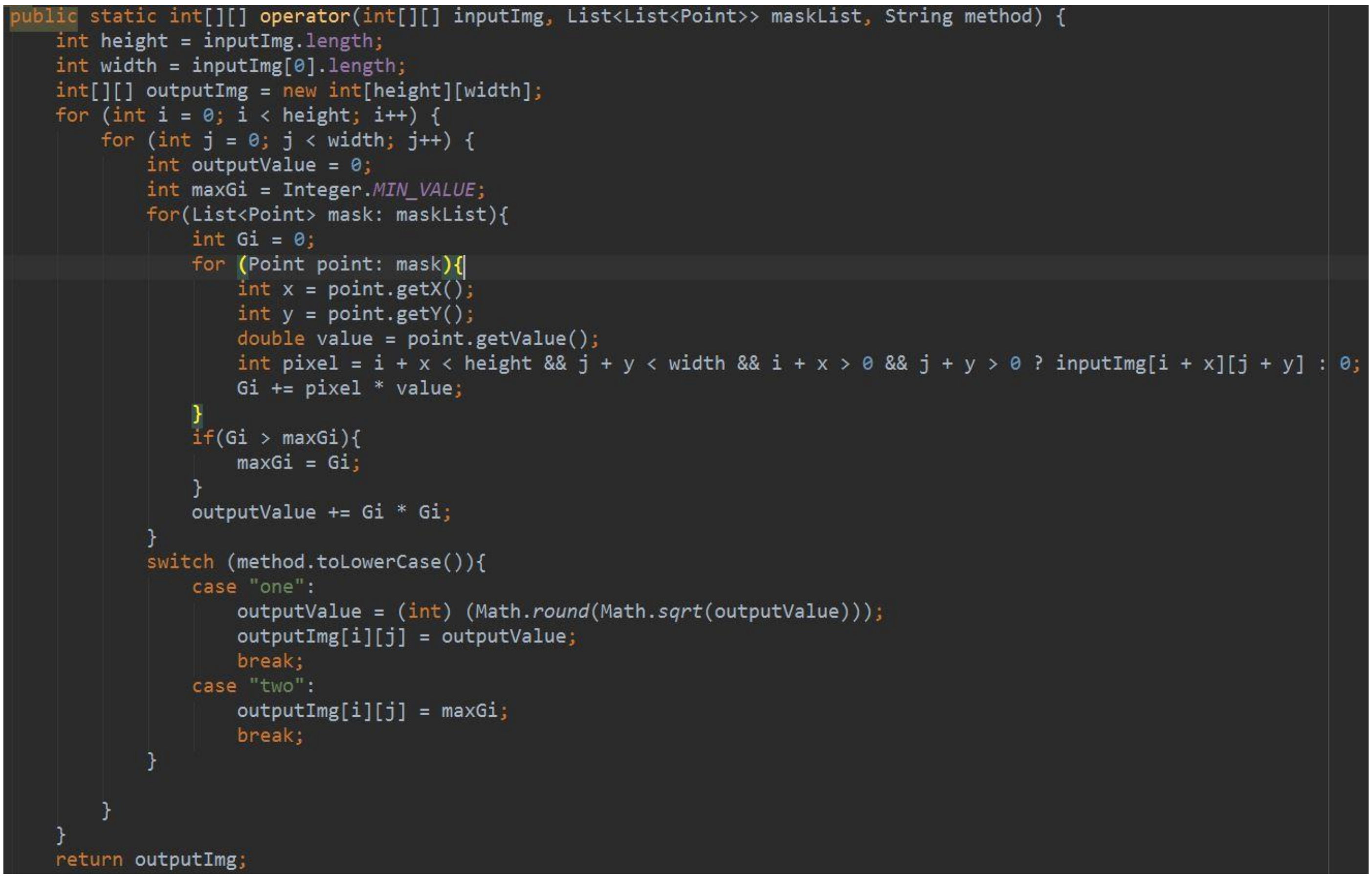


```java
System.out.println("Doing Kirsch edge detection...");
maskList = new ArrayList<>();
maskList.add(Mask.getKirschZero());
maskList.add(Mask.getKirschOne());
maskList.add(Mask.getKirschTwo());
maskList.add(Mask.getKirschThree());
maskList.add(Mask.getKirschFour());
maskList.add(Mask.getKirschFive());
maskList.add(Mask.getKirschSix());
maskList.add(Mask.getKirschSeven());
outputImg =  EdgeDetection.operator(img, maskList, "two");
outputImg = ImageProcess.binaryImage(outputImg, 350);
FileProcess.outputImg(outputImg, "kirch.bmp");

System.out.println("Doing Robinsion edge detection...");
maskList = new ArrayList<>();
maskList.add(Mask.getRobinsonZero());
maskList.add(Mask.getRobinsonOne());
maskList.add(Mask.getRobinsonTwo());
maskList.add(Mask.getRobinsonThree());
maskList.add(Mask.getRobinsonFour());
maskList.add(Mask.getRobinsonFive());
maskList.add(Mask.getRobinsonSix());
maskList.add(Mask.getRobinsonSeven());
outputImg =  EdgeDetection.operator(img, maskList, "two");
outputImg = ImageProcess.binaryImage(outputImg, 100);
FileProcess.outputImg(outputImg, "robinson.bmp");
```



```java
System.out.println("Doing Nevatia edge detection...");
maskList = new ArrayList<>();
maskList.add(Mask.getNevatiaZero());
maskList.add(Mask.getNevatiaThirty());
maskList.add(Mask.getNevatiaNegativeThirty());
maskList.add(Mask.getNevatiaSixty());
maskList.add(Mask.getNevatiaNegativeSixty());
maskList.add(Mask.getNevatiaNegativeNinety());
outputImg =  EdgeDetection.operator(img, maskList, "two");
outputImg = ImageProcess.binaryImage(outputImg, 20000);
FileProcess.outputImg(outputImg, "nevatia.bmp");
```

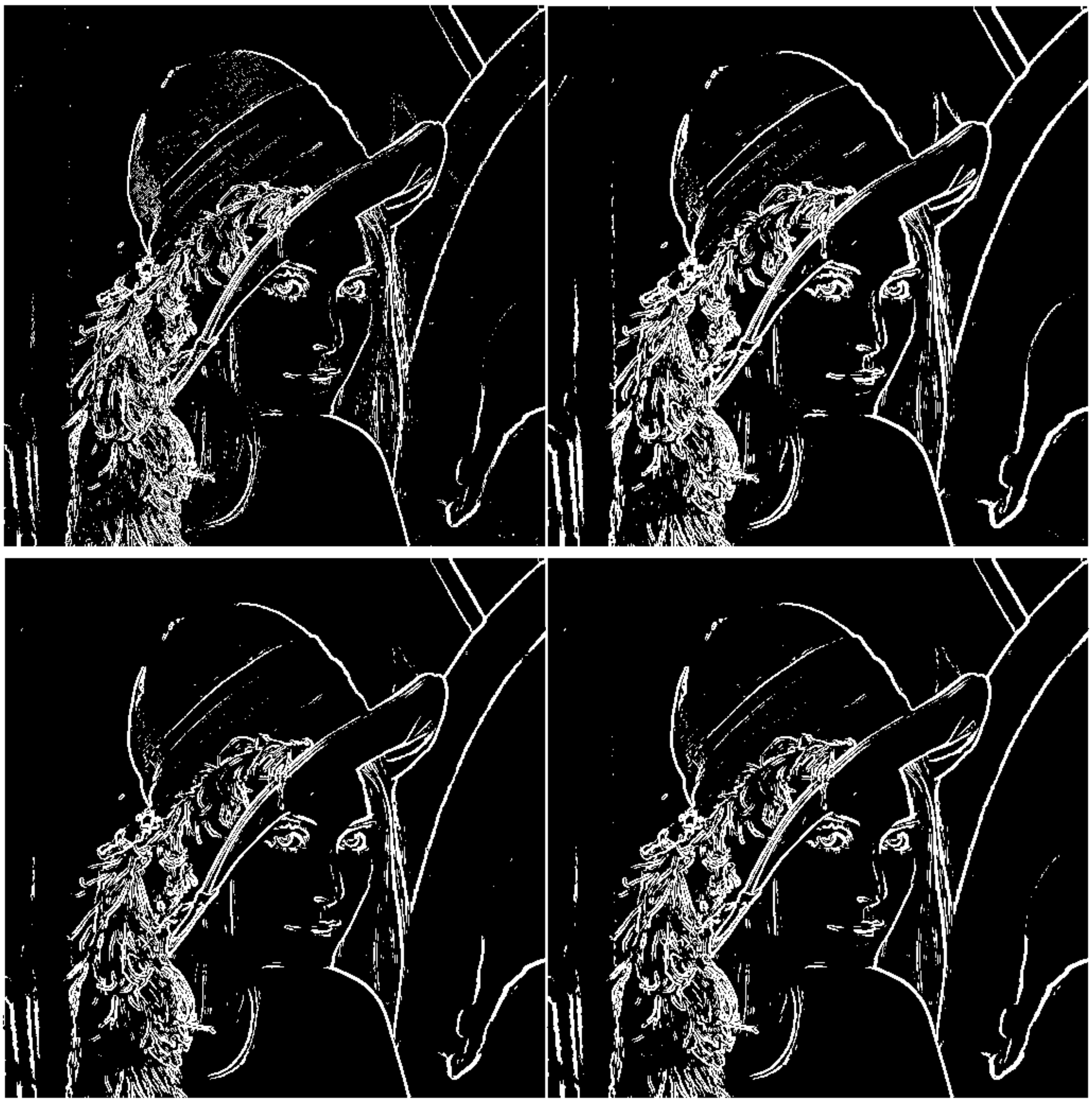
Edge detection operator(./Source Code/src/hw9/EdgeDetection.java):



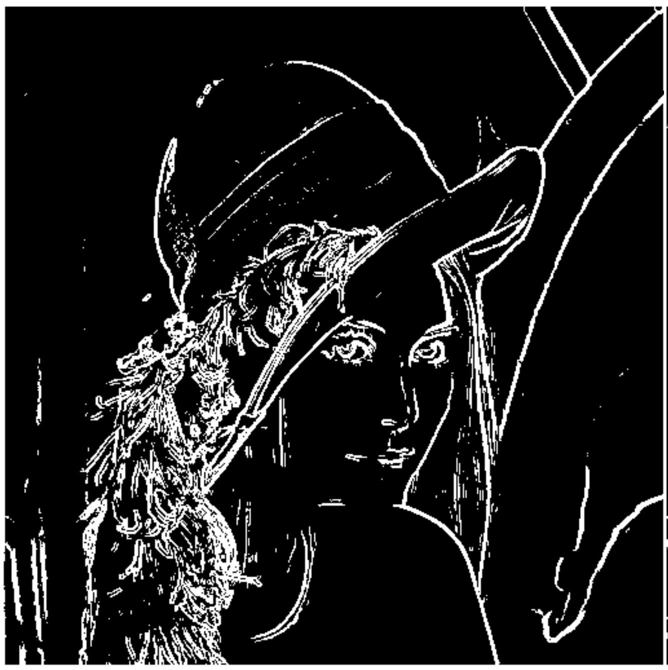
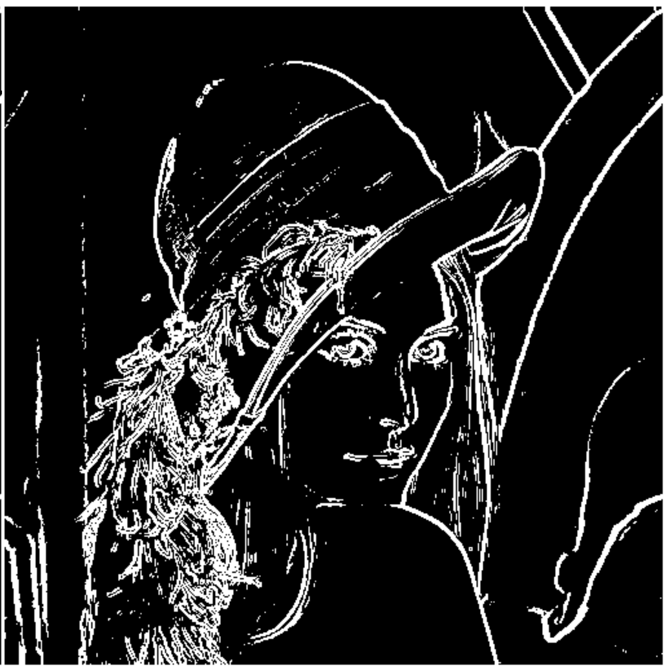
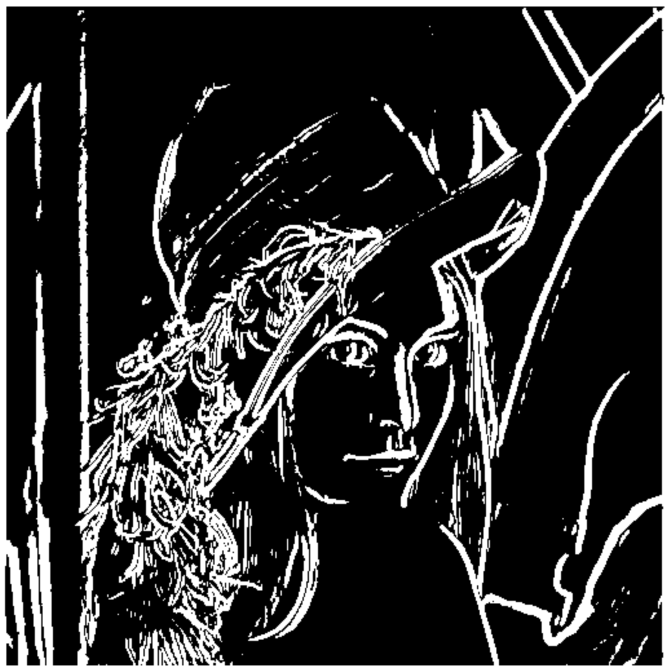
```java
public static int[][] operator(int[][] inputImg, List<List<Point>> maskList, String method) {
    int height = inputImg.length;
    int width = inputImg[0].length;
    int[][] outputImg = new int[height][width];
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            int outputValue = 0;
            int maxGi = Integer.MIN_VALUE;
            for(List<Point> mask: maskList){
                int Gi = 0;
                for (Point point: mask){
                    int x = point.getX();
                    int y = point.getY();
                    double value = point.getValue();
                    int pixel = i + x < height && j + y < width && i + x > 0 && j + y > 0 ? inputImg[i + x][j + y] : 0;
                    Gi += pixel * value;
                }
                if(Gi > maxGi){
                    maxGi = Gi;
                }
                outputValue += Gi * Gi;
            }
            switch (method.toLowerCase()){
                case "one":
                    outputValue = (int) (Math.round(Math.sqrt(outputValue)));
                    outputImg[i][j] = outputValue;
                    break;
                case "two":
                    outputImg[i][j] = maxGi;
                    break;
            }

        }
    }
    return outputImg;
```

4. 執行結果：

Robert(Threshold:25), Prewitt(75), Sobel(125), Frei and Chen(100):



Kirsch(Threshold: 400), Robinson(100), Nevatia-Babu(20000):

5. 如何執行

執行./source code/ hw9jar 即可產生作業要求的兩種圖片(lena.bmp 需要和 hw9.jar 在同一個資料夾底下)

程式進入點為 ./Surce Code/src/hw9/MainEdgeDetection.java