Computer vision
Homework9
General Edge Detection

B06902091 資工四 羅寶瑩

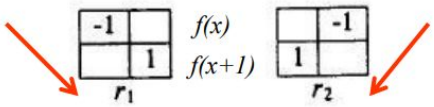
Description

You are to implement following edge detectors with thresholds :

To implement each operator, we follow the method introduced in the lecture slide.

(a) Robert's Operator

$f'(x) \approx f(x+1) - f(x)$



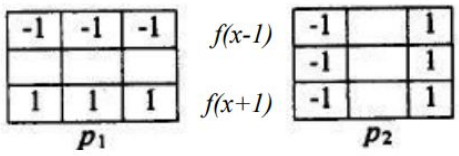
gradient magnitude: $\sqrt{r_1^2 + r_2^2}$

```
int r1 = ori.at<uchar>(i + 1, j + 1) - ori.at<uchar>(i, j);
int r2 = ori.at<uchar>(i + 1, j) - ori.at<uchar>(i, j + 1);
int gradient = sqrt(r1 * r1 + r2 * r2);
if (gradient >= threshold) {
    output.at<uchar>(i, j) = 0;
}
else {
    output.at<uchar>(i, j) = 255;
}
```

(For more information, please refer to lines 14~28.)

(b) Prewitt's Edge Detector

$f'(x) \approx f(x+1) - f(x-1)$



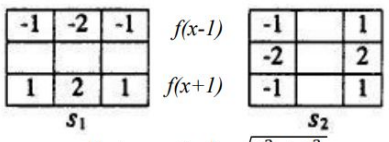
gradient magnitude: $\sqrt{p_1^2 + p_2^2}$

```
int p1 = (array[0][0] + array[0][1] + array[0][2]) * -1 + array[2][0] + array[2][1] + array[2][2];
int p2 = (array[0][0] + array[1][0] + array[2][0]) * -1 + array[0][2] + array[1][2] + array[2][2];
int gradient = sqrt(p1 * p1 + p2 * p2);
if (gradient >= threshold) {
    output.at<uchar>(i, j) = 0;
}
else {
    output.at<uchar>(i, j) = 255;
}
```

(For more information, please refer to lines 30~97.)

(c) Sobel's Edge Detector

$f'(x) \approx f(x+1) - f(x-1)$

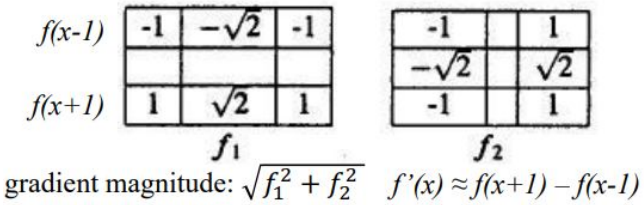


gradient magnitude: $\sqrt{s_1^2 + s_2^2}$

```
int s1 = - array[0][0] - (array[0][1]*2) - array[0][2] + array[2][0] + (array[2][1]*2) + array[2][2];
int s2 = - array[0][0] - (array[1][0]*2) - array[2][0] + array[0][2] + (array[1][2]*2) + array[2][2];
int gradient = sqrt(s1 * s1 + s2 * s2);
if (gradient >= threshold) {
    output.at<uchar>(i, j) = 0;
}
else {
    output.at<uchar>(i, j) = 255;
}
```

(For more information, please refer to lines 99~166.)

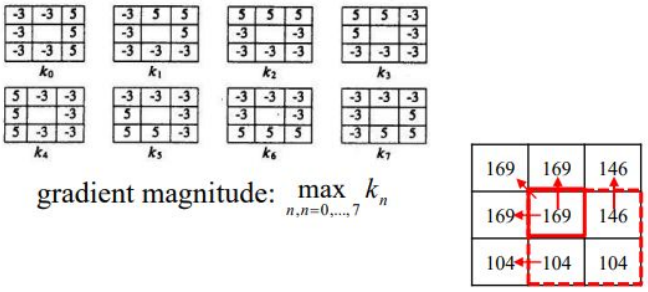
(d) Frei and Chen's Gradient Operator




```
int f1 = -array[0][0] - (array[0][1] * sqrt(2)) - array[0][2] + array[2][0] + (array[2][1] * sqrt(2)) + array[2][2];
int f2 = -array[0][0] - (array[1][0] * sqrt(2)) - array[2][0] + array[0][2] + (array[1][2] * sqrt(2)) + array[2][2];
int gradient = sqrt(f1 * f1 + f2 * f2);
if (gradient >= threshold) {
    output.at<uchar>(i, j) = 0;
}
else {
    output.at<uchar>(i, j) = 255;
}
```

(For more information, please refer to lines 168~235.)

(e) Kirsch's Compass Operator

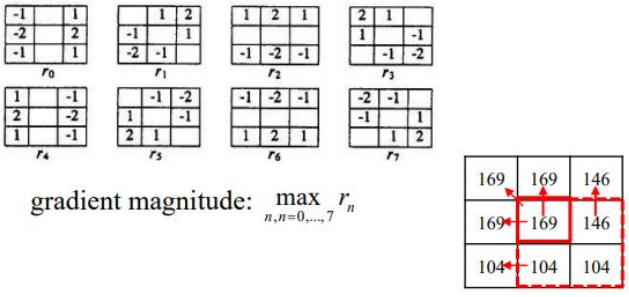



```
int k[8];
//array[][]
k[0] = (-3) * (array[0][0] + array[0][1] + array[1][0] + array[2][0] + array[2][1]) + (5) * (array[0][2] + array[1][2] + array[2][2]);
k[1] = (-3) * (array[0][0] + array[1][0] + array[2][0] + array[2][1] + array[2][2]) + (5) * (array[0][1] + array[0][2] + array[1][2]);
k[2] = (-3) * (array[1][0] + array[1][2] + array[2][0] + array[2][1] + array[2][2]) + (5) * (array[0][0] + array[0][1] + array[0][2]);
k[3] = (-3) * (array[2][0] + array[2][1] + array[2][2] + array[1][2] + array[0][2]) + (5) * (array[0][0] + array[0][1] + array[1][0]);
k[4] = (-3) * (array[0][1] + array[0][2] + array[2][1] + array[2][2] + array[1][2]) + (5) * (array[0][0] + array[1][0] + array[2][0]);
k[5] = (-3) * (array[0][0] + array[0][1] + array[0][2] + array[1][2] + array[2][2]) + (5) * (array[1][0] + array[2][0] + array[2][1]);
k[6] = (-3) * (array[0][0] + array[0][1] + array[0][2] + array[1][0] + array[1][2]) + (5) * (array[2][0] + array[2][1] + array[2][2]);
k[7] = (-3) * (array[0][0] + array[0][1] + array[0][2] + array[1][0] + array[2][0]) + (5) * (array[2][1] + array[2][2] + array[1][2]);


int gradient = -1;
for (int a = 0; a < 8; a++) {
    if (k[a] > gradient) {
        gradient = k[a];
    }
}
if (gradient >= threshold) {
    output.at<uchar>(i, j) = 0;
}
else {
    output.at<uchar>(i, j) = 255;
}
```

(For source code, please refer to lines 237~319.)

(f) Robinson's Compass Operator



(For source code, please refer to lines 321~402.)

(g) Nevatia-Babu 5x5 Operator

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°

gradient magnitude: $\max_{n,n=0,\ldots,5} N_n$

(For source code, please refer to lines 404~491.)

Result

(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

Reference:

1. *lecture slide*