Computer Vision
Homework 9 Report

林義聖
B03902048

December 19, 2016

1 Introduction

In this homework assignment, we're going to do **General Edge Detection**. I use *Python* as my programming language and *Pillow* as my image library. It is a fork of *PIL*, which is the original image library of *Python*. And I use *Pillow* for reading input image, and transfering image data into a *List* of *Python*.

Figure 1: lena.bmp as Benchmark Image

2 General Edge Detection

To do edge detection, I apply each kernel and calculate the gradient magnitude g for each operator. When the kernel reach the boundary of the image, I use a function to flip the inward pixels to the outward.

```
def flip(p, boundary):
  if p < 0:
    return -p -1
  elif p >= boundary:
    return 2 * boundary - p - 1
  else:
    return p
```

2.1 Robert's Operator

Gradient magnitude of this operator is $g = \sqrt{r_1^2 + r_2^2}$.

```
def roberts_operator(data, hei, wid):
  r1 = [\
    ( 0, 0,-1),( 1, 0, 0),\
    ( 0, 1, 0),( 1, 1, 1)]
  r2 = [\
    ( 0, 0, 0),( 1, 0,-1),\
    ( 0, 1, 1),( 1, 1, 0)]
  ret = []
  for y in range(hei):
    for x in range(wid):
      r1val = 0
      for rx, ry, val in r1:
        px = flip(x + rx, wid)
        py = flip(y + ry, hei)
        r1val += data[py * wid + px] * val
      r2val = 0
      for rx, ry, val in r2:
        px = flip(x + rx, wid)
        py = flip(y + ry, hei)
        r2val += data[py * wid + px] * val
      val = sqrt(r1val ** 2 + r2val ** 2)
      ret.append(0 if val >= 12 else 255)
  return ret
```

Figure 2: Robert's operator with threshold = 12

2.2 Prewitt's Edge Detector

The method to calculate gradient magnitude of this operator is the same as previous one, and the kernel of this operator is,

```
p1 = [\
  (-1,-1,-1),( 0,-1,-1),( 1,-1,-1),\
  (-1, 0, 0),( 0, 0, 0),( 1, 0, 0),\
  (-1, 1, 1),( 0, 1, 1),( 1, 1, 1)]
p2 = [\
  (-1,-1,-1),( 0,-1, 0),( 1,-1, 1),\
  (-1, 0,-1),( 0, 0, 0),( 1, 0, 1),\
  (-1, 1,-1),( 0, 1, 0),( 1, 1, 1)]
```

Figure 3: Prewitt's edge detector with threshold = 24

2.3 Sobel's Edge Detector

The method to calculate gradient magnitude of this operator is the same as previous one, and the kernel of this operator is,

```
s1 = [\
  (-1,-1,-1),( 0,-1,-2),( 1,-1,-1),\
  (-1, 0, 0),( 0, 0, 0),( 1, 0, 0),\
  (-1, 1, 1),( 0, 1, 2),( 1, 1, 1)]
s2 = [\
  (-1,-1,-1),( 0,-1, 0),( 1,-1, 1),\
  (-1, 0,-2),( 0, 0, 0),( 1, 0, 2),\
  (-1, 1,-1),( 0, 1, 0),( 1, 1, 1)]
```

Figure 4: Sobel's edge detector with threshold = 38

2.4 Frei and Chen's Gradient Operator

The method to calculate gradient magnitude of this operator is the same as previous one, and the kernel of this operator is,

```
f1 = [\
  (-1,-1,-1),( 0,-1,-sqrt(2)),( 1,-1,-1),\
  (-1, 0, 0),( 0, 0,        0),( 1, 0, 0),\
  (-1, 1, 1),( 0, 1, sqrt(2)),( 1, 1, 1)]
f2 = [\
  (-1,-1,       -1),( 0,-1, 0),( 1,-1,        1),\
  (-1, 0,-sqrt(2)),( 0, 0, 0),( 1, 0, sqrt(2)),\
  (-1, 1,       -1),( 0, 1, 0),( 1, 1,        1)]
```

Figure 5: Frei and Chen's gradient operator with threshold = 30

2.5 Kirsch's Compass Operator

Gradient magnitude of this operator is $g = \max_{n,n=0,\ldots,7} k_n$.

```
def kirsch_operator(data, hei, wid):
  k = [
    [\
    (-1,-1,-3),( 0,-1,-3),( 1,-1, 5),\
    (-1, 0,-3),( 0, 0, 0),( 1, 0, 5),\
    (-1, 1,-3),( 0, 1,-3),( 1, 1, 5)],\
    [\
    (-1,-1,-3),( 0,-1, 5),( 1,-1, 5),\
    (-1, 0,-3),( 0, 0, 0),( 1, 0, 5),\
    (-1, 1,-3),( 0, 1,-3),( 1, 1,-3)],\
    [\
    (-1,-1, 5),( 0,-1, 5),( 1,-1, 5),\
    (-1, 0,-3),( 0, 0, 0),( 1, 0,-3),\
    (-1, 1,-3),( 0, 1,-3),( 1, 1,-3)],\
    [\
    (-1,-1, 5),( 0,-1, 5),( 1,-1,-3),\
    (-1, 0, 5),( 0, 0, 0),( 1, 0,-3),\
    (-1, 1,-3),( 0, 1,-3),( 1, 1,-3)],\
    [\
    (-1,-1, 5),( 0,-1,-3),( 1,-1,-3),\
    (-1, 0, 5),( 0, 0, 0),( 1, 0,-3),\
    (-1, 1, 5),( 0, 1,-3),( 1, 1,-3)],\
    [\
    (-1,-1,-3),( 0,-1,-3),( 1,-1,-3),\
    (-1, 0, 5),( 0, 0, 0),( 1, 0,-3),\
    (-1, 1, 5),( 0, 1, 5),( 1, 1,-3)],\
    [\
    (-1,-1,-3),( 0,-1,-3),( 1,-1,-3),\
    (-1, 0,-3),( 0, 0, 0),( 1, 0,-3),\
    (-1, 1, 5),( 0, 1, 5),( 1, 1, 5)],\
    [\
    (-1,-1,-3),( 0,-1,-3),( 1,-1,-3),\
    (-1, 0,-3),( 0, 0, 0),( 1, 0, 5),\
```

```
    (-1, 1,-3),( 0, 1, 5),( 1, 1, 5)]\
  ]
  ret = []
  for y in range(hei):
    for x in range(wid):
      maxval = -float('Inf')
      for kernel in k:
        tmpval = 0
        for rx, ry, val in kernel:
          px = flip(x + rx, wid)
          py = flip(y + ry, hei)
          tmpval += data[py * wid + px] * val
        maxval = tmpval if tmpval > maxval else maxval
      ret.append(0 if maxval >= 135 else 255)
  return ret
```

Figure 6: Kirsch's compass operator with threshold = 135

2.6 Robinson's Compass Operator

The method to calculate gradient magnitude of this operator is the same as previous one, and the kernel of this operator is,

```
k = [
  [\
  (-1,-1,-1),( 0,-1, 0),( 1,-1, 1),\
  (-1, 0,-2),( 0, 0, 0),( 1, 0, 2),\
  (-1, 1,-1),( 0, 1, 0),( 1, 1, 1)],\
  [\
  (-1,-1, 0),( 0,-1, 1),( 1,-1, 2),\
  (-1, 0,-1),( 0, 0, 0),( 1, 0, 1),\
  (-1, 1,-2),( 0, 1,-1),( 1, 1, 0)],\
  [\
  (-1,-1, 1),( 0,-1, 2),( 1,-1, 1),\
  (-1, 0, 0),( 0, 0, 0),( 1, 0, 0),\
  (-1, 1,-1),( 0, 1,-2),( 1, 1,-1)],\
  [\
  (-1,-1, 2),( 0,-1, 1),( 1,-1, 0),\
```

```
    (-1, 0, 1),( 0, 0, 0),( 1, 0,-1),\
    (-1, 1, 0),( 0, 1,-1),( 1, 1,-2)],\
    [\
    (-1,-1, 1),( 0,-1, 0),( 1,-1,-1),\
    (-1, 0, 2),( 0, 0, 0),( 1, 0,-2),\
    (-1, 1, 1),( 0, 1, 0),( 1, 1,-1)],\
    [\
    (-1,-1, 0),( 0,-1,-1),( 1,-1,-2),\
    (-1, 0, 1),( 0, 0, 0),( 1, 0,-1),\
    (-1, 1, 2),( 0, 1, 1),( 1, 1, 0)],\
    [\
    (-1,-1,-1),( 0,-1,-2),( 1,-1,-1),\
    (-1, 0, 0),( 0, 0, 0),( 1, 0, 0),\
    (-1, 1, 1),( 0, 1, 2),( 1, 1, 1)],\
    [\
    (-1,-1,-2),( 0,-1,-1),( 1,-1, 0),\
    (-1, 0,-1),( 0, 0, 0),( 1, 0, 1),\
    (-1, 1, 0),( 0, 1, 1),( 1, 1, 2)]\
]
```

Figure 7: Robinson's compass operator with threshold = 43

2.7 Nevatia-Babu 5×5 Operator

The method to calculate gradient magnitude of this operator is the same as previous one, and the kernel of this operator is,

```
k = [
    [\
    (-2,-2, 100),(-1,-2, 100),( 0,-2, 100),( 1,-2, 100),( 2,-2, 100),\
    (-2,-1, 100),(-1,-1, 100),( 0,-1, 100),( 1,-1, 100),( 2,-1, 100),\
    (-2, 0,   0),(-1, 0,   0),( 0, 0,   0),( 1, 0,   0),( 2, 0,   0),\
    (-2, 1,-100),(-1, 1,-100),( 0, 1,-100),( 1, 1,-100),( 2, 1,-100),\
    (-2, 2,-100),(-1, 2,-100),( 0, 2,-100),( 1, 2,-100),( 2, 2,-100)],\
    [\
    (-2,-2, 100),(-1,-2, 100),( 0,-2, 100),( 1,-2, 100),( 2,-2, 100),\
    (-2,-1, 100),(-1,-1, 100),( 0,-1, 100),( 1,-1,  78),( 2,-1, -32),\
    (-2, 0, 100),(-1, 0,  92),( 0, 0,   0),( 1, 0, -92),( 2, 0,-100),\
```

```
12    (-2, 1,  32),(-1, 1, -78),( 0, 1,-100),( 1, 1,-100),( 2, 1,-100),\
13    (-2, 2,-100),(-1, 2,-100),( 0, 2,-100),( 1, 2,-100),( 2, 2,-100)],\
14    [\
15    (-2,-2, 100),(-1,-2, 100),( 0,-2, 100),( 1,-2,  32),( 2,-2,-100),\
16    (-2,-1, 100),(-1,-1, 100),( 0,-1,  92),( 1,-1, -78),( 2,-1,-100),\
17    (-2, 0, 100),(-1, 0, 100),( 0, 0,   0),( 1, 0,-100),( 2, 0,-100),\
18    (-2, 1, 100),(-1, 1,  78),( 0, 1, -92),( 1, 1,-100),( 2, 1,-100),\
19    (-2, 2, 100),(-1, 2, -32),( 0, 2,-100),( 1, 2,-100),( 2, 2,-100)],\
20    [\
21    (-2,-2,-100),(-1,-2,-100),( 0,-2,   0),( 1,-2, 100),( 2,-2, 100),\
22    (-2,-1,-100),(-1,-1,-100),( 0,-1,   0),( 1,-1, 100),( 2,-1, 100),\
23    (-2, 0,-100),(-1, 0,-100),( 0, 0,   0),( 1, 0, 100),( 2, 0, 100),\
24    (-2, 1,-100),(-1, 1,-100),( 0, 1,   0),( 1, 1, 100),( 2, 1, 100),\
25    (-2, 2,-100),(-1, 2,-100),( 0, 2,   0),( 1, 2, 100),( 2, 2, 100)],\
26    [\
27    (-2,-2,-100),(-1,-2,  32),( 0,-2, 100),( 1,-2, 100),( 2,-2, 100),\
28    (-2,-1,-100),(-1,-1, -78),( 0,-1,  92),( 1,-1, 100),( 2,-1, 100),\
29    (-2, 0,-100),(-1, 0,-100),( 0, 0,   0),( 1, 0, 100),( 2, 0, 100),\
30    (-2, 1,-100),(-1, 1,-100),( 0, 1, -92),( 1, 1,  78),( 2, 1, 100),\
31    (-2, 2,-100),(-1, 2,-100),( 0, 2,-100),( 1, 2, -32),( 2, 2, 100)],\
32    [\
33    (-2,-2, 100),(-1,-2, 100),( 0,-2, 100),( 1,-2, 100),( 2,-2, 100),\
34    (-2,-1, -32),(-1,-1,  78),( 0,-1, 100),( 1,-1, 100),( 2,-1, 100),\
35    (-2, 0,-100),(-1, 0, -92),( 0, 0,   0),( 1, 0,  92),( 2, 0, 100),\
36    (-2, 1,-100),(-1, 1,-100),( 0, 1,-100),( 1, 1, -78),( 2, 1,  32),\
37    (-2, 2,-100),(-1, 2,-100),( 0, 2,-100),( 1, 2,-100),( 2, 2,-100)]\
38  ]
```

Figure 8: Nevatia-Babu $5x5$ operator with threshold = 12500

3 How to Use

There's only one program `edge_detection.py`, to use it, just type command "`./edge_detection.py --operator=[operator] [input file]`".