Computer Vision
Homework 10 Report

林義聖
B03902048

December 23, 2016

1 Introduction

In this homework assignment, we're going to do **Zero Crossing 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 Zero Crossing Edge Detection

To do zero crossing edge detection, I apply each kernel on image data. When the kernel reach the boundary of the image, I use a function to flip the inward pixels to the outward.

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

And to apply the kernel on the given image data, I write a function to do convolution with given kernel, data and weight:

```
1  def conv_runner(data, hei, wid, kernel,
       weight):
2    ret = []
3    for y in range(hei):
4      for x in range(wid):
5        s = 0
6        for kx, ky, val in kernel:
7          px = flip(x + kx, wid)
8          py = flip(y + ky, hei)
9          s += data[py * wid + px] * val
10       ret.append(s * weight)
11   return ret
```

2.1 Laplacian

All of my kernel in the program is in this form, `(offset_x, offset_y, weight)`. And the following is the kernel and weight of Laplacian zero crossing edge detector.

```
k = [\
  (-1,-1, 0),( 0,-1, 1),( 1,-1, 0),\
  (-1, 0, 1),( 0, 0,-4),( 1, 0, 1),\
  (-1, 1, 0),( 0, 1, 1),( 1, 1, 0)\
]
weight = 1
```

Figure 2: Laplacian with threshold = 15

2.2 Minimum-Variance Laplacian

The following is the kernel and weight.

```
k = [\
   (-1,-1, 2),( 0,-1,-1),( 1,-1, 2),\
   (-1, 0,-1),( 0, 0,-4),( 1, 0,-1),\
   (-1, 1, 2),( 0, 1,-1),( 1, 1, 2)\
]
weight = 1/3
```

Figure 3: Minimum-variance with threshold = 20

```
   (-5, 5, 0),(-4, 5, 0),(-3, 5,  0),(-2,
        5, -1),(-1, 5, -1),( 0, 5, -2),( 1,
         5, -1),( 2, 5, -1),( 3, 5,  0),(
        4, 5, 0),( 5, 5, 0)\
]
weight = 1
```

Figure 4: LOG with threshold = 3000

2.3 Laplacian Of Gaussian

The following is the kernel and weight.

```
k = [\
   (-5,-5, 0),(-4,-5, 0),(-3,-5,  0)
        ,(-2,-5, -1),(-1,-5, -1),( 0,-5,
        -2),( 1,-5, -1),( 2,-5, -1),( 3,-5,
          0),( 4,-5, 0),( 5,-5, 0),\
   (-5,-4, 0),(-4,-4, 0),(-3,-4, -2)
        ,(-2,-4, -4),(-1,-4, -8),( 0,-4,
        -9),( 1,-4, -8),( 2,-4, -4),( 3,-4,
         -2),( 4,-4, 0),( 5,-4, 0),\
   (-5,-3, 0),(-4,-3,-2),(-3,-3, -7)
        ,(-2,-3,-15),(-1,-3,-22),(
        0,-3,-23),( 1,-3,-22),( 2,-3,-15),(
         3,-3, -7),( 4,-3,-2),( 5,-3, 0),\
   (-5,-2,-1),(-4,-2,-4),(-3,-2,-15)
        ,(-2,-2,-24),(-1,-2,-14),( 0,-2,
        -1),( 1,-2,-14),( 2,-2,-24),(
        3,-2,-15),( 4,-2,-4),( 5,-2,-1),\
   (-5,-1,-1),(-4,-1,-8),(-3,-1,-22)
        ,(-2,-1,-14),(-1,-1, 52),(
        0,-1,103),( 1,-1, 52),( 2,-1,-14),(
         3,-1,-22),( 4,-1,-8),( 5,-1,-1),\
   (-5, 0,-2),(-4, 0,-9),(-3, 0,-23),(-2,
        0, -1),(-1, 0,103),( 0, 0,178),( 1,
         0,103),( 2, 0, -1),( 3, 0,-23),(
        4, 0,-9),( 5, 0,-2),\
   (-5, 1,-1),(-4, 1,-8),(-3, 1,-22),(-2,
        1,-14),(-1, 1, 52),( 0, 1,103),( 1,
         1, 52),( 2, 1,-14),( 3, 1,-22),(
        4, 1,-8),( 5, 1,-1),\
   (-5, 2,-1),(-4, 2,-4),(-3, 2,-15),(-2,
        2,-24),(-1, 2,-14),( 0, 2, -1),( 1,
         2,-14),( 2, 2,-24),( 3, 2,-15),(
        4, 2,-4),( 5, 2,-1),\
   (-5, 3, 0),(-4, 3,-2),(-3, 3, -7),(-2,
        3,-15),(-1, 3,-22),( 0, 3,-23),( 1,
         3,-22),( 2, 3,-15),( 3, 3, -7),(
        4, 3,-2),( 5, 3, 0),\
   (-5, 4, 0),(-4, 4, 0),(-3, 4, -2),(-2,
        4, -4),(-1, 4, -8),( 0, 4, -9),( 1,
         4, -8),( 2, 4, -4),( 3, 4, -2),(
        4, 4, 0),( 5, 4, 0),\
```

2.4 Difference of Gaussian

The following is the core program to generate the kernel.

```
def gaussian(x, mu, sig):
  return (1 / np.sqrt(2*np.power(sig,2)*
      np.pi) ) * np.exp(-np.power(x - mu,
       2.) / (2 * np.power(sig, 2.)))

def gauss_2d(x, y, mu_x, mu_y, sig_x,
    sig_y):
  return gaussian(x, mu_x, sig_x) *
      gaussian(y, mu_y, sig_y)

def DOG(x, y):
  return gauss_2d(x, y, 0, 0, 1, 1) -
      gauss_2d(x, y, 0, 0, 3, 3)

k = []
for y in range(-5, 6):
  for x in range(-5, 6):
    k.append((x, y, DOG(x, y)))
```

Figure 5: DOG with threshold = 1

3 How to Use

There's only one program `zero_crossing.py`, to know how to use it, just type command "`./zero_crossing.py -h`". And to use it, type command "`./zero_crossing.py -k [kernel] -t [threshold] [input image]`".