Computer Vision hw10

B05902050

黃子源

I use PIL to complete the homework. In my program, I use function getpixel() and putpixel() to get the value of every pixel.

I create a list named mask to represent the 3x3 block coordinate which the original is in the center.

the thresholds I used:

Laplace mask 1: 15

Laplace mask 2: 15

Minimum variance Laplacian: 20

Laplace of Gaussian: 3000

Difference of Gaussian: 10000

Laplace mask 1 and 2:

I calculate the sum of pixel value of the neighbor of every pixel with 4-connected and 8-connected.

Laplace mask 1 = The sum of 4-connected – 4*(original pixel vale)

Laplace mask 2 = (The sum of 8-connected – 8*(original pixel vale))/3

If the mask value is larger than threshold, then take the pixel as an edge.

Principal code fragment:

```
from PIL import Image
import numpy as np
import math

img = (Image.open("lena.bmp")).convert("L")
w, h = img.size
laplace1 = Image.new("1", (w, h))
laplace2 = Image.new("1", (w, h))
for i in range(1,w-1):
    for j in range(1,h-1):
        connect4 = 0
        connect8 = 0
        for k in range(-1,2):
            for l in range(-1,2):
                if (k == 0 or l == 0) and (k != l):
                    connect4 += img.getpixel((i+l,j+k))
                    connect8 += img.getpixel((i+l,j+k))
                elif (k,l) != (0,0):
                    connect8 += img.getpixel((i+l,j+k))
        connect4 -= 4*img.getpixel((i,j))
        connect8 -= 8*img.getpixel((i,j))
        connect8 /= 3
        if connect4 > 15:
            laplace1.putpixel((i,j),0)
        else:
            laplace1.putpixel((i,j),1)
        if connect8 > 15:
            laplace2.putpixel((i,j),0)
        else:
            laplace2.putpixel((i,j),1)
laplace1.save('laplace1.bmp')
laplace2.save('laplace2.bmp')
```

Minimum variance Laplacian:

I use the 3x3 mask to calculate the minimum variance digital Laplacian of every pixel.

If it is larger than threshold, take the pixel as an edge pixel.

Principal code fragment:

```
minvarlaplace = Image.new("1", (w, h))
mask = [(-1,-1),(0,-1),(1,-1),(-1,0),(0,0),(1,0),(-1,1),(0,1),(1,1)]
for i in range(1,w-1):
    for j in range(1,h-1):
        p = [0 for i in range(9)]
        for k in range(9):
            p[k] =  img.getpixel((i + mask[k][0],j + mask[k][1]))
        minvar = 2*(p[0] + p[2] + p[6] + p[8]) - (p[1] + p[3] + p[5] + p[7]) - 4*p[4]
        minvar /= 3
        if minvar > 20:
            minvarlaplace.putpixel((i,j),0)
        else:
            minvarlaplace.putpixel((i,j),1)
minvarlaplace.save('minvarlaplace.bmp')
```

Laplacian of Gaussian and Difference of Gaussian:

The calculation of this part is quite complicated, so I use brute force to construct a 11x11 mask, so there's no any technical method in this part of my code. Note that the threshold value of Difference of Gaussian is opposite because unlike other method, the edge of this part has smaller mask

Principal code fragment:

```
guassianlap = Image.new("1", (w,h))
differencelap = Image.new("1", (w,h))
for i in range(5,w-5):
    for j in range(5,h-5):
        p = [[0]*11 for i in range(11)]
        for k in range(11):
            for l in range(11):
                p[l][k] = img.getpixel((i+l-5,j+k-5))
        guassian = 178*p[5][5] + 103*(p[5][4] + p[5][6] + p[4][5] + p[6][5]) - 1*(p[5][3] + p[5][7] + p[3][5] + p[7]
        guassian2 = 283*p[5][5] + 160*(p[5][4] + p[5][6] + p[4][5] + p[6][5]) + 15*(p[5][3] + p[5][7] + p[3][5] + p[7
        if guassian > 3000:
            guassianlap.putpixel((i,j),0)
        else:
            guassianlap.putpixel((i,j),1)
        if guassian2 < 10000:
            differencelap.putpixel((i,j),0)
        else:
            differencelap.putpixel((i,j),1)
guassianlap.save('guassianlap.bmp')
differencelap.save('differencelap.bmp')
```

Result:

Laplace mask 1:

Laplace mask 2:

Minimum variance Laplacian:

Laplacian of Gaussian:

Difference of Gaussian: