

# 922 U0610 電腦視覺 Computer Vision

## Homework 10

授課教師： 傅楸善 教授

學生系級： 資工所一年級

學生姓名： 姚嘉昇

學生學號： R06922002

# I. INTRODUCTION

## 1.1. Descriptions of Problem

This homework is to do zero crossing edge detection with following rules:

- A. Laplacian mask 1 with threshold of 15.
- B. Laplacian mask 2 with threshold of 15.
- C. Minimum variance Laplacian with threshold of 20.
- D. Laplacian of Gaussian with threshold of 3000.
- E. Difference of Gaussian with threshold of 1.

## 1.2. Programming Tools

- 1.2.1. Programming Language: Python3
- 1.2.2. Programming IDE: Visual Studio Code

## II. METHOD

### 2.1. Algorithms

#### 2.1.1. Laplacian mask 1

	1	
1	-4	1
	1	

#### 2.1.2. Laplacian mask 2

$\frac{1}{3}$	1	1	1
	1	-8	1
	1	1	1

#### 2.1.3. Minimum variance Laplacian

$\frac{1}{3}$	2	-1	2
	-1	-4	-1
	2	-1	2

#### 2.1.4. Laplacian of Gaussian

0	0	0	-1	-1	-2	-1	-1	0	0	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	-2	-7	-15	-22	-23	-22	-15	-7	-2	0
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-2	-9	-23	-1	103	178	103	-1	-23	-9	-2
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
0	-2	-7	-15	-22	-23	-22	-15	-7	-2	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	0	0	-1	-1	-2	-1	-1	0	0	0

#### 2.1.5. Difference of Gaussian

-1	-3	-4	-6	-7	-8	-7	-6	-4	-3	-1
-3	-5	-8	-11	-13	-13	-13	-11	-8	-5	-3
-4	-8	-12	-16	-17	-17	-17	-16	-12	-8	-4
-6	-11	-16	-16	0	15	0	-16	-16	-11	-6
-7	-13	-17	0	85	160	85	0	-17	-13	-7
-8	-13	-17	15	160	283	160	15	-17	-13	-8
-7	-13	-17	0	85	160	85	0	-17	-13	-7
-6	-11	-16	-16	0	15	0	-16	-16	-11	-6
-4	-8	-12	-16	-17	-17	-17	-16	-12	-8	-4
-3	-5	-8	-11	-13	-13	-13	-11	-8	-5	-3
-1	-3	-4	-6	-7	-8	-7	-6	-4	-3	-1

## 2.2. Code Fragments

### 2.2.1. Code fragments of this homework

```

1 def getLaplacianMask1Array(originalImage, threshold):
2     """
3     :type originalImage: Image (from PIL)
4     :type threshold: float
5     :return type: numpy array
6     """
7     from PIL import Image
8     import numpy as np
9     # Zero numpy array with the same size.
10    LaplacianMask1 = np.zeros(originalImage.size)
11    # Scan each column in original image.
12    for c in range(originalImage.size[0]):
13        # Scan each row in original image.
14        for r in range(originalImage.size[1]):
15            # Calculate x0, y0, x1, y1, x2, y2 and avoid out of image range.
16            x0 = max(c - 1, 0)
17            y0 = max(r - 1, 0)
18            x1 = c
19            y1 = r
20            x2 = min(c + 1, originalImage.size[0] - 1)
21            y2 = min(r + 1, originalImage.size[1] - 1)
22            # Get 3x3 neighbors.
23            neighbors = [originalImage.getpixel((x0, y0)), originalImage.getpixel((x1, y0)), originalImage.getpixel((x2, y0)),
24                        originalImage.getpixel((x0, y1)), originalImage.getpixel((x1, y1)), originalImage.getpixel((x2, y1)),
25                        originalImage.getpixel((x0, y2)), originalImage.getpixel((x1, y2)), originalImage.getpixel((x2, y2))]
26            # Calculate Gradient magnitude of Laplacian mask 1.
27            magnitude = (0) * neighbors[0] + (1) * neighbors[1] + (0) * neighbors[2] + \
28                        (1) * neighbors[3] + (-4) * neighbors[4] + (1) * neighbors[5] + \
29                        (0) * neighbors[6] + (1) * neighbors[7] + (0) * neighbors[8]
30            # Binarize with threshold.
31            if (magnitude >= threshold):
32                LaplacianMask1[c, r] = 1
33            elif (magnitude <= -threshold):
34                LaplacianMask1[c, r] = -1
35            else:
36                LaplacianMask1[c, r] = 0
37    return LaplacianMask1

```

Figure 2.2.1.1. Code of Laplacian mask 1.

```

39 def getLaplacianMask2Array(originalImage, threshold):
40     """
41     :type originalImage: Image (from PIL)
42     :type threshold: float
43     :return type: numpy array
44     """
45     from PIL import Image
46     import numpy as np
47     # Zero numpy array with the same size.
48     LaplacianMask2 = np.zeros(originalImage.size)
49     # Scan each column in original image.
50     for c in range(originalImage.size[0]):
51         # Scan each row in original image.
52         for r in range(originalImage.size[1]):
53             # Calculate x0, y0, x1, y1, x2, y2 and avoid out of image range.
54             x0 = max(c - 1, 0)
55             y0 = max(r - 1, 0)
56             x1 = c
57             y1 = r
58             x2 = min(c + 1, originalImage.size[0] - 1)
59             y2 = min(r + 1, originalImage.size[1] - 1)
60             # Get 3x3 neighbors.
61             neighbors = [originalImage.getpixel((x0, y0)), originalImage.getpixel((x1, y0)), originalImage.getpixel((x2, y0)),
62                         originalImage.getpixel((x0, y1)), originalImage.getpixel((x1, y1)), originalImage.getpixel((x2, y1)),
63                         originalImage.getpixel((x0, y2)), originalImage.getpixel((x1, y2)), originalImage.getpixel((x2, y2))]
64             # Calculate Gradient magnitude of Laplacian mask 2.
65             magnitude = (1) * neighbors[0] + (1) * neighbors[1] + (1) * neighbors[2] + \
66                         (1) * neighbors[3] + (-8) * neighbors[4] + (1) * neighbors[5] + \
67                         (1) * neighbors[6] + (1) * neighbors[7] + (1) * neighbors[8]
68             magnitude = magnitude / 3
69             # Binarize with threshold.
70             if (magnitude >= threshold):
71                 LaplacianMask2[c, r] = 1
72             elif (magnitude <= -threshold):
73                 LaplacianMask2[c, r] = -1
74             else:
75                 LaplacianMask2[c, r] = 0
76     return LaplacianMask2

```

Figure 2.2.1.2. Code of Laplacian mask 2.

```

78 def getMinVarianceLaplacianArray(originalImage, threshold):
79     """
80     :type originalImage: Image (from PIL)
81     :type threshold: float
82     :return type: numpy array
83     """
84     from PIL import Image
85     import numpy as np
86     # Zero numpy array with the same size.
87     minVarianceLaplacian = np.zeros(originalImage.size)
88     # Scan each column in original image.
89     for c in range(originalImage.size[0]):
90         # Scan each row in original image.
91         for r in range(originalImage.size[1]):
92             # Calculate x0, y0, x1, y1, x2, y2 and avoid out of image range.
93             x0 = max(c - 1, 0)
94             y0 = max(r - 1, 0)
95             x1 = c
96             y1 = r
97             x2 = min(c + 1, originalImage.size[0] - 1)
98             y2 = min(r + 1, originalImage.size[1] - 1)
99             # Get 3x3 neighbors.
100            neighbors = [originalImage.getpixel((x0, y0)), originalImage.getpixel((x1, y0)), originalImage.getpixel((x2, y0)),
101            originalImage.getpixel((x0, y1)), originalImage.getpixel((x1, y1)), originalImage.getpixel((x2, y1)),
102            originalImage.getpixel((x0, y2)), originalImage.getpixel((x1, y2)), originalImage.getpixel((x2, y2))]
103            # Calculate Gradient magnitude of Laplacian mask 2.
104            magnitude = (2) * neighbors[0] + (-1) * neighbors[1] + (2) * neighbors[2] + \
105            (-1) * neighbors[3] + (-4) * neighbors[4] + (-1) * neighbors[5] + \
106            (2) * neighbors[6] + (-1) * neighbors[7] + (2) * neighbors[8]
107            magnitude = magnitude / 3
108            # Binarize with threshold.
109            if (magnitude >= threshold):
110                minVarianceLaplacian[c, r] = 1
111            elif (magnitude <= -threshold):
112                minVarianceLaplacian[c, r] = -1
113            else:
114                minVarianceLaplacian[c, r] = 0
115        return minVarianceLaplacian

```

Figure 2.2.1.3. Code of Minimum variance Laplacian.

```

117 def getLaplacianOfGaussianArray(originalImage, threshold):
118     """
119     :type originalImage: Image (from PIL)
120     :type threshold: float
121     :return type: numpy array
122     """
123     from PIL import Image
124     import numpy as np
125     # Kernel of Laplacian of Gaussian.
126     kernel = [ [0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0],
127                [0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
128                [0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
129                [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
130                [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
131                [-2, -9, -23, -1, 103, 178, 103, -1, -23, -9, -2],
132                [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
133                [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
134                [0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
135                [0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
136                [0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0]]
137     # Zero numpy array with the same size.
138     LaplacianOfGaussianArray = np.zeros(originalImage.size)
139     # Scan each column in original image.
140     for c in range(originalImage.size[0]):
141         # Scan each row in original image.
142         for r in range(originalImage.size[1]):
143             # Calculate x0-10, y0-10 and avoid out of image range.
144             x = np.zeros(11)
145             y = np.zeros(11)
146             for i in range(11):
147                 x[i] = np.clip(c + (i - 5), 0, originalImage.size[0] - 1)
148                 y[i] = np.clip(r + (i - 5), 0, originalImage.size[1] - 1)
149             # Get 11x11 neighbors.
150             neighbors = np.zeros((11, 11))
151             for i in range(11):
152                 for j in range(11):
153                     neighbors[i, j] = originalImage.getpixel((x[i], y[j]))
154             # Calculate Gradient magnitude of Laplacian of Gaussian.
155             magnitude = 0
156             for i in range(11):
157                 for j in range(11):
158                     magnitude = magnitude + kernel[j][i] * neighbors[i, j]
159             # Binarize with threshold.
160             if (magnitude >= threshold):
161                 LaplacianOfGaussianArray[c, r] = 1
162             elif (magnitude <= -threshold):
163                 LaplacianOfGaussianArray[c, r] = -1
164             else:
165                 LaplacianOfGaussianArray[c, r] = 0
166     return LaplacianOfGaussianArray

```

Figure 2.2.1.4. Code of Laplacian of Gaussian.

```

168 def getDifferenceOfGaussianArray(originalImage, threshold):
169     """
170     :type originalImage: Image (from PIL)
171     :type threshold: float
172     :return type: numpy array
173     """
174     from PIL import Image
175     import numpy as np
176     # Kernel of Laplacian of Gaussian.
177     kernel = [ [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1],
178                [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
179                [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
180                [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
181                [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
182                [-8, -13, -17, 15, 160, 283, 160, 15, -17, -13, -8],
183                [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
184                [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
185                [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
186                [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
187                [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1]]
188     # Zero numpy array with the same size.
189     DifferenceOfGaussianArray = np.zeros(originalImage.size)
190     # Scan each column in original image.
191     for c in range(originalImage.size[0]):
192         # Scan each row in original image.
193         for r in range(originalImage.size[1]):
194             # Calculate x0-10, y0-10 and avoid out of image range.
195             x = np.zeros(11)
196             y = np.zeros(11)
197             for i in range(11):
198                 x[i] = np.clip(c + (i - 5), 0, originalImage.size[0] - 1)
199                 y[i] = np.clip(r + (i - 5), 0, originalImage.size[1] - 1)
200             # Get 11x11 neighbors.
201             neighbors = np.zeros((11, 11))
202             for i in range(11):
203                 for j in range(11):
204                     neighbors[i, j] = originalImage.getpixel((x[i], y[j]))
205             # Calculate Gradient magnitude of Difference of Gaussian.
206             magnitude = 0
207             for i in range(11):
208                 for j in range(11):
209                     magnitude = magnitude + kernel[j][i] * neighbors[i, j]
210             # Binarize with threshold.
211             if (magnitude >= threshold):
212                 DifferenceOfGaussianArray[c, r] = 1
213             elif (magnitude <= -threshold):
214                 DifferenceOfGaussianArray[c, r] = -1
215             else:
216                 DifferenceOfGaussianArray[c, r] = 0
217     return DifferenceOfGaussianArray

```

Figure 2.2.1.5. Code of Difference of Gaussian.



```
219 def zeroCrossingDetector(grandient, width, height):
220     """
221     :type grandient: numpy array
222     :type width: int
223     :type height: int
224     :return type: Image (from PIL)
225     """
226     # New image with the same size and 'binary' format.
227     zeroCrossingImage = Image.new('1', grandient.shape)
228     # Scan each column in grandient array.
229     for c in range(grandient.shape[0]):
230         # Scan each row in grandient array.
231         for r in range(grandient.shape[1]):
232             # Record does it cross zero.
233             cross = 1
234             # If current location is high.
235             if (grandient[c, r] == 1):
236                 # Scan its neighbors.
237                 for x in range(-width // 2, width // 2 + 1):
238                     for y in range(-height // 2, height // 2 + 1):
239                         # Avoid out of range.
240                         destX = np.clip(c + x, 0, grandient.shape[0] - 1)
241                         destY = np.clip(r + y, 0, grandient.shape[1] - 1)
242                         # Check zero crossing.
243                         if (grandient[destX, destY] == -1):
244                             cross = 0
245             # Put pixel to image.
246             zeroCrossingImage.putpixel((c, r), cross)
247     return zeroCrossingImage
```

Figure 2.2.1.6. Code of zero crossing detector.

## III. RESULTS

### 3.1. Original Image



Figure 3.1. Original lena.bmp.

### 3.2. Result of Laplacian Mask 1 with Threshold of 15



Figure 3.2.1. Original lena.bmp.



Figure 3.2.2. Laplacian Mask 1.bmp.

### 3.3. Result of Laplacian Mask 2 with Threshold of 15



Figure 3.3.1. Original lena.bmp.



Figure 3.3.2. Laplacian Mask 2.bmp.

### 3.4. Result of Minimum Variance Laplacian with Threshold of 20



Figure 3.4.1. Original lena.bmp.



Figure 3.4.2. min-Variance Laplacian.bmp.

### 3.5. Result of Laplacian of Gaussian with Threshold of 3000



Figure 3.5.1. Original lena.bmp.



Figure 3.5.2. Laplacian of Gaussian.bmp.

### 3.6. Result of Difference of Gaussian with Threshold of 1



Figure 3.6.1. Original lena.bmp.



Figure 3.6.2. Difference of Gaussian.bmp.