Computer Vision Homework #10

資工四 b05902115 陳建丞

Original

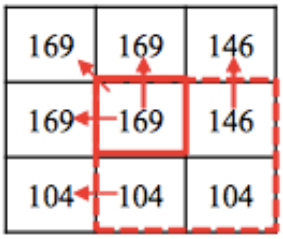
Result

Laplacian1 (Threshold = 15)	Laplacian2 (Threshold = 15)	Min-Variance Laplacian (Threshold = 38)
		

Laplacian of Gaussian (Threshold = 3000)	Difference of Gaussian (Threshold = 1)
	

Implementation

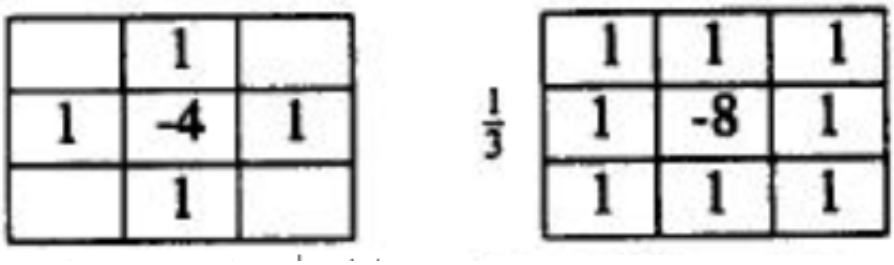
- **Border Padding**
  - To handle the border condition, I write a function `Border_padding(img, size)` to return a padded version of the input image. The padding method is the same as

    The padding size depends on the filter size of different operators.
- **Edge Dectection Operator**

  All the operators follow the same processing steps

  1. Create corresponding filter (Laplacian、Minimum-Variance Laplacian、LOG、DOG)
  2. Border padding according to filter size
  3. Convolutionally traverse through each pixel
  4. Compute gradient magnitude
  5. Compare the result with threshold and -threshold.
     - If gradient magnitude $\geq$ threshold, mark as 1
     - If gradient magnitude $\leq$ -threshold, mark as -1
     - else, mark as 0
  6. Store the marked result to a marked image.
  7. Traverse through the marked image to apply zero crossing.
  8. If a pixel's marked value is 1 and any of its 8 neighbors is -1. $\Rightarrow$ edge pixel
- **Laplacian**



- **Minimum-Variance Laplacian**

$$\frac{1}{3}\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

- **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

- **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

Code Segment

- Border Padding

```
def Border_padding(img, size)

  height, width = img.shape
  win = int(size/2)
  r = win*2
  new_img = np.zeros((height+r, width+r), dtype=int)
  new_height, new_width = new_img.shape

  for i in range(height):
    for j in range(width):
      new_img[i+win][j+win] = img[i][j]

  for i in range(height):
    new_img[i+win, :win] = img[i][0]
    new_img[i+win, -win:] = img[i][width-1]
  for i in range(width):
    new_img[:win, i+win] = img[0][i]
    new_img[-win:, i+win] = img[height-1][i]
  new_img[:win, :win] = img[0][0]
  new_img[:win, -win:] = img[0][width-1]
  new_img[-win:, :win] = img[height-1][0]
  new_img[-win:, -win:] = img[height-1][width-1]
  return new_img
```

- Zero Crossing

```
  # Zero Crossing
  for i in range(height):
    for j in range(width):
      new_img[i][j] = 255
      if marked_img[i][j] != 1:
        continue

      cross = False
      for r in [-1, 0, 1]:
        for c in [-1, 0, 1]:
          if i+r < 0 or i+r >= height or j+c < 0 or j+c >= width or cross:
            continue
          if marked_img[i+r][j+c] == -1:
            new_img[i][j] = 0
            cross = True
```