Computer Vision — Homework 4

B05902118 **陳盈如**

October 9, 2020

1 Dilation

The definition of *Dilation* is as follows.

$$A \oplus B = \{c \in E^N | c = a + b, a \in A, b \in B\}$$

Then, here is the code.

```
4  def dilation(img, kernel):
5      dil = np.zeros(img.shape, int)
6      for i in range(img.shape[0]):
7          for j in range(img.shape[1]):
8              if img[i][j] != 0:
9                  for dot in kernel:
10                     if i + dot[0] >= 0 and i + dot[0] < img.shape[0] \
11                     and j + dot[1] >= 0 and j + dot[1] < img.shape[1]:
12                         dil[i + dot[0]][j + dot[1]] = 255
13     return dil
```

Result:

(a) Original (b) Result

Figure 1: Dilation

2 Erosion

The definition of *Erosion* is as follows.

$$A \ominus B = \{x \in E^N | (x + b) \in A, b \in B\}$$

All B will be contained in A. Then, here is the code.

```
15  def erosion(img, kernel):
16      ero = np.zeros(img.shape, int)
17      for i in range(img.shape[0]):
18          for j in range(img.shape[1]):
19              ok = 1
20              for dot in kernel:
21                  if i + dot[0] < 0 or i + dot[0] >= img.shape[0] \
22                  or j + dot[1] < 0 or j + dot[1] >= img.shape[1] \
23                  or img[i + dot[0]][j + dot[1]] == 0:
24                      ok = 0
25                      break
26              if ok:
27                  ero[i][j] = 255
28      return ero
```

Result:

(a) Original (b) Result

Figure 2: Erosion

3 Opening

The definition of *Opening* is as follows.

$$A \circ B = (A \ominus B) \oplus B$$

. Do *Erosion* first, then do *Dilation*.

Result:

(a) Original (b) Result

Figure 3: Opening

4 Closing

The definition of *Closing* is as follows. It's contrary to *Opening*.

$$A \bullet B = (A \oplus B) \ominus B$$

Result:

(a) Original (b) Result

Figure 4: Closing

5 Hit-and-miss transform

The definition of *Hit-and-miss transform* is as follows.

$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$$

Here is the code.

```
30  def Hit_and_Miss(img, J_kernel, K_kernel):
31      imgC = np.zeros(img.shape, int)
32      imgC = -img + 255
33      cv2.imwrite('img_comp.bmp', imgC)
34      J = erosion(img, J_kernel)
35      K = erosion(imgC, K_kernel)
36      ham = np.zeros(img.shape, int)
37      for i in range(img.shape[0]):
38          for j in range(img.shape[1]):
39              if J[i][j] != 0 and K[i][j] != 0:
40                  ham[i][j] = 255
41      return ham
```

Result:

(a) Original (b) Hit-and-miss transform