電腦視覺 HW5 report

D10922012 資工所 王傳啟

Write programs which do gray-scale morphology on a gray-scale image(lena.bmp):

(a)~(d) 所共用的 kernel, 存放為需要處理的座標, 相對應到原點之位置 與 課程中提到規定的 value 值 0。表示法: (相對 x, 相對 y, 值 v)

```
    # kernel ([x, y, value])
    kernel = np.array(
        [
            [-2, -1,  0], [-2,  0,  0], [-2,  1,  0],
            [-1, -2,  0], [-1, -1,  0], [-1,  0,  0], [-1,  1,  0], [-
1,  2,  0],
            [ 0, -2,  0], [ 0, -1,  0], [ 0,  0,  0], [ 0,  1,  0],
[ 0,  2,  0],
            [ 1, -2,  0], [ 1, -1,  0], [ 1,  0,  0], [ 1,  1,  0],
[ 1,  2,  0],
            [ 2, -1,  0], [ 2,  0,  0], [ 2,  1,  0]
        ]
    )
```

(a) Dilation:

採取講義之公式(如下), 對每個 pixel 進行下數之運作

$$(f \oplus k)(x) = \max\{f(x-z) + k(z) | z \in K, x - z \in F\}$$

```
def dilation(np_img, kernel):
    row = np_img.shape[0]
    col = np_img.shape[1]
    dilation_img = np.zeros((row, col), dtype=int)
    for i in range(row):
        for j in range(col):
            max_value = 0 # max 暫存值
            for k in kernel:
                tmp = 0
                new_i = i - k[0] # 公式 f(x-z) 的 x 座標
                new_j = j - k[1] # 公式 f(x-z) 的 y 座標
                # 若在原圖邊界內, 就運行公式 f(x-z) + k(z)
                if new_i >= 0 and new_i < row and \
                    new_j >= 0 and new_j < col:
                    tmp = np_img[new_i][new_j]+k[2]
```

```
                    if max_value < tmp: # 原公式，取最大值的步驟
                        max_value = tmp
            dilation_img[i][j] = max_value
    return dilation_img
```

(b) Erosion:

採取講義之公式(如下)，對每個 pixel 進行下數之運作

$$(f \ominus k)(x) = \min\{f(x+z) - k(z)\}$$

```
def erosion(np_img, kernel):
    row = np_img.shape[0]
    col = np_img.shape[1]
    erosion_img = np.zeros((row, col), dtype=int)
    for i in range(row):
        for j in range(col):
            min_value = 256
            for k in kernel:
                tmp = 0
                new_i = i + k[0] # 公式 f(x+z)  的 x 座標
                new_j = j + k[1] # 公式 f(x+z)  的 y 座標
                 # 若在原圖邊界內，就運行公式 f(x+z) - k(z)
                if new_i >= 0 and new_i < row and \
                    new_j >= 0 and new_j < col:
                    tmp = np_img[new_i][new_j] - k[2]
                    if tmp < min_value: #原公式，取最小值的步驟
                        min_value = tmp
            if min_value < 0: # 若小於 0，則校正為 0
```

```
                min_value = 0
            erosion_img[i][j] = min_value
    return erosion_img
```

(c) Opening:

Opening 就是對原圖執行 kernel 的 Erosion 再做 Dilation

```
np_img_c = dilation(erosion(np_img, kernel), kernel)
```

(d) Closing:

Closing 就是對原圖執行 kernel 的 Dilation 再做 Erosion

```
np_img_d = erosion(dilation(np_img, kernel), kernel)
```