Computer Vision I

Homework 4 - Mathematical Morphology - Binary B04902090 資工四 施長元

Usage of the full code:

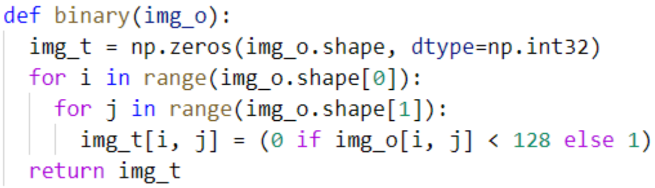
```
python main.py [Image_path]
```

After the code exit, the output files will be in the same directory with the main.py

Environment: Python3.6 on Windows Linux Subsystem (Ubuntu 16.04)

Contents:

Before we begin, binarize the picture first.



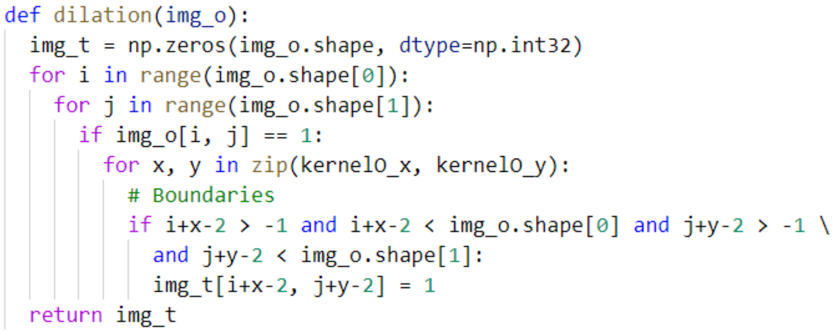
```
def binary(img_o):
    img_t = np.zeros(img_o.shape, dtype=np.int32)
    for i in range(img_o.shape[0]):
        for j in range(img_o.shape[1]):
            img_t[i, j] = (0 if img_o[i, j] < 128 else 1)
    return img_t
```

Kernel: Octagonal 3-5-5-5-3 kernel

Write programs which do binary morphological:

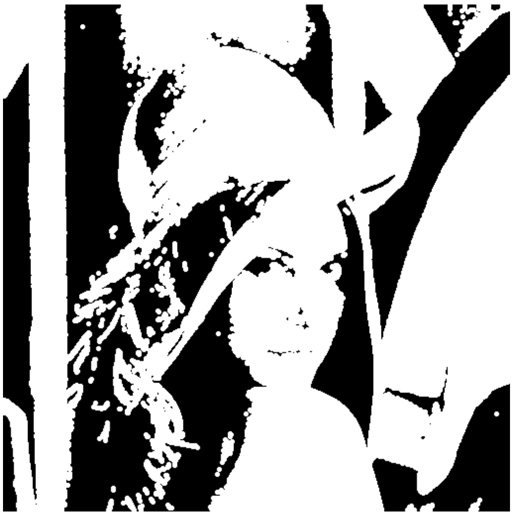
- Dilation

  兩層迴圈跑過所有 pixel，倘若此 pixel 是 1，套用 kernel 在上面，並將 kernel 上所有(除了超出圖片的)pixel 設為 1，輸出時*255

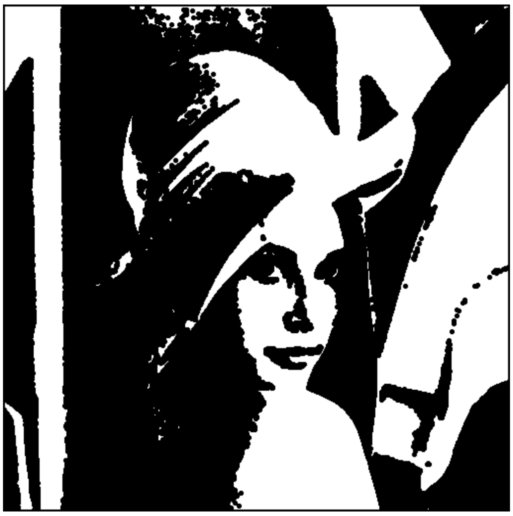


```
def dilation(img_o):
    img_t = np.zeros(img_o.shape, dtype=np.int32)
    for i in range(img_o.shape[0]):
        for j in range(img_o.shape[1]):
            if img_o[i, j] == 1:
                for x, y in zip(kernel0_x, kernel0_y):
                    # Boundaries
                    if i+x-2 > -1 and i+x-2 < img_o.shape[0] and j+y-2 > -1 \
                        and j+y-2 < img_o.shape[1]:
                        img_t[i+x-2, j+y-2] = 1
    return img_t
```

Results: 左圖



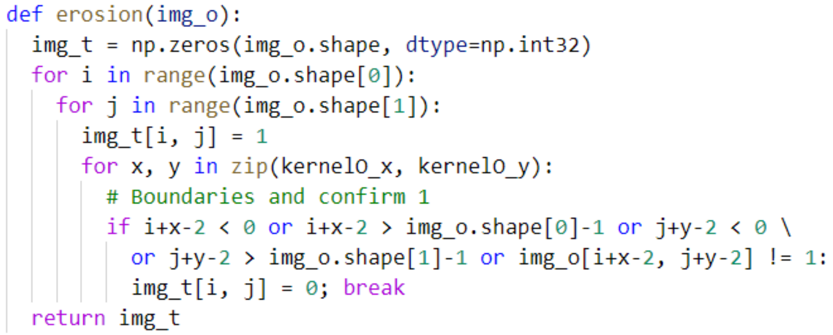
▲Dilation



▲Erosion

- Erosion

  兩層迴圈跑過所有 pixel，套用 kernel 在上面，倘若 kernel 上有任何 pixel 為 0，或是超出 boundaries，則將此 pixel 設為 0，輸出時*255

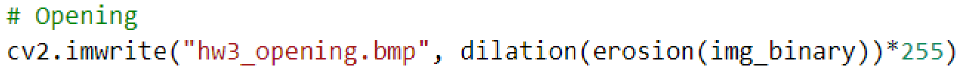


```python
def erosion(img_o):
    img_t = np.zeros(img_o.shape, dtype=np.int32)
    for i in range(img_o.shape[0]):
        for j in range(img_o.shape[1]):
            img_t[i, j] = 1
            for x, y in zip(kernelO_x, kernelO_y):
                # Boundaries and confirm 1
                if i+x-2 < 0 or i+x-2 > img_o.shape[0]-1 or j+y-2 < 0 \
                    or j+y-2 > img_o.shape[1]-1 or img_o[i+x-2, j+y-2] != 1:
                    img_t[i, j] = 0; break
    return img_t
```

  Result: 上頁圖右側

- Opening

  先 erosion 再 dilation ($B \circ K = (B \ominus K) \oplus K$)，輸出時*255

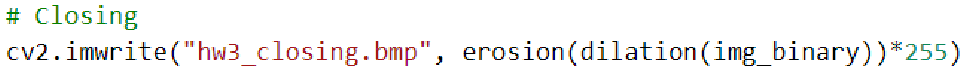


```python
# Opening
cv2.imwrite("hw3_opening.bmp", dilation(erosion(img_binary))*255)
```

  Result: 下圖左側

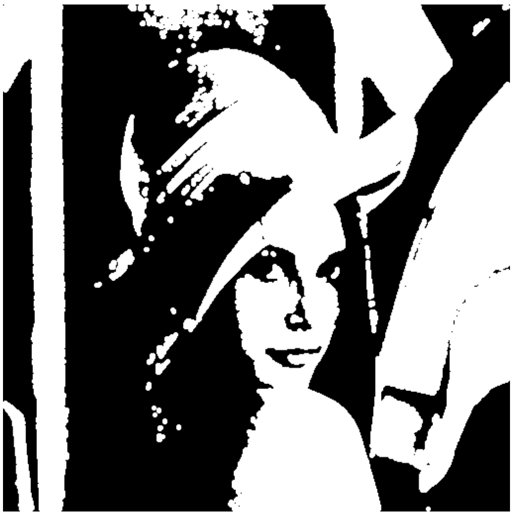
- Closing

  先 dilation 再 erosion ($B \cdot K = (B \oplus K) \ominus K$) ，輸出時*255

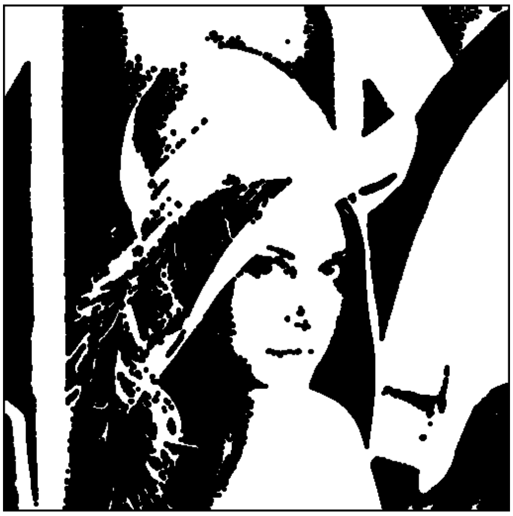


```python
# Closing
cv2.imwrite("hw3_closing.bmp", erosion(dilation(img_binary))*255)
```

  Result: 右側



▲Opening

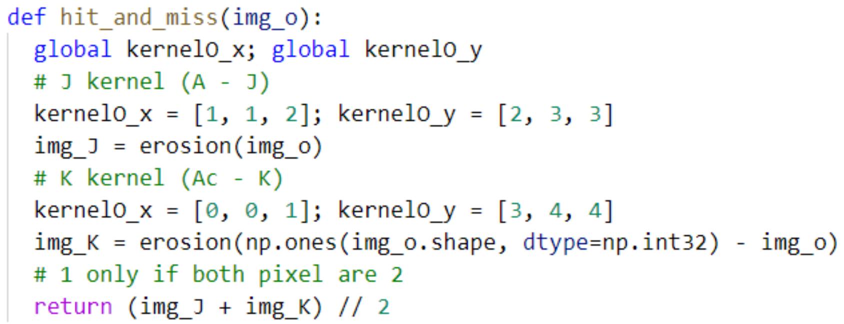


▲Closing

- Hit-and-miss transform

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

  用 binarized image 與 L shaped kernel 做 erosion,，然後用 inversed binarized image 與 L shaped kernel 往右上移一單位做 erosion，然後把兩個結果做聯集，輸出時*255



```python
def hit_and_miss(img_o):
    global kernelO_x; global kernelO_y
    # J kernel (A - J)
    kernelO_x = [1, 1, 2]; kernelO_y = [2, 3, 3]
    img_J = erosion(img_o)
    # K kernel (Ac - K)
    kernelO_x = [0, 0, 1]; kernelO_y = [3, 4, 4]
    img_K = erosion(np.ones(img_o.shape, dtype=np.int32) - img_o)
    # 1 only if both pixel are 2
    return (img_J + img_K) // 2
```

Result: