HW4	4109037012 資工四 李曉旻

Coding part :

threshold 對一個灰階圖像進行二值化處理。先創建了一個與原始圖像大小相同的零矩陣，再將原始圖像中大於閾值的像素點在矩陣中設置為 maxval 值。可以將圖像中的像素值簡化為 0 或 maxval 兩個值，方便進一步的處理和分析。

```
def threshold(img, thresh, maxval):
    if len(img.shape) > 2:
        raise ValueError("Input image should be grayscale")

    thresholded = np.zeros_like(img)
    thresholded[img > thresh] = maxval
    return thresholded
```

這個 function 實做了獲取形態學操作的結構元素的功能。它根據指定的形狀(矩形或十字形)和大小，生成對應的結構元素。對於矩形元素，直接使用全 1 矩陣，而對於十字形元素，在中心行和列上設置為 1，其餘為 0。生成的結構元素可用於腐蝕、膨脹等操作。

```
def getStructuringElement(shape, ksize):
    if shape == cv2.MORPH_RECT:
        return np.ones(ksize, dtype=np.uint8)
    elif shape == cv2.MORPH_CROSS:
        kernel = np.zeros(ksize, dtype=np.uint8)
        center = (ksize[0] // 2, ksize[1] // 2)
        kernel[center[0], :] = 1
        kernel[:, center[1]] = 1
        return kernel
    else:
        raise ValueError("Unsupported shape")
```

這個 function 實做了圖像的腐蝕操作。首先對輸入圖像進行填充，然後使用雙重循環遍歷圖像的每個像素，檢查以其為中心的鄰域是否與結構元素完全匹配，並將該像素值保留在輸出圖像中。利用這種方式，可以消除圖像中的小雜點和毛刺，使圖像更加平滑。

```
def erode(img, kernel):
    h, w = img.shape[:2]
    kh, kw = kernel.shape[:2]
    pad_h, pad_w = (kh - 1) // 2, (kw - 1) // 2
    padded_img = np.pad(img, ((pad_h, pad_h), (pad_w, pad_w)),
mode='constant')
    eroded_img = np.zeros_like(img)
    for i in range(h):
        for j in range(w):
```

```
            if np.all(padded_img[i:i+kh, j:j+kw] == img[i, j]):
                eroded_img[i, j] = img[i, j]
    return eroded_img
```

這個 function 實做了圖像的膨脹操作。與上個 function 一樣使用雙重循環遍歷圖像的每個像素，檢查以其為中心的鄰域是否與結構元素有任意重疊，並將該像素值設為 255(白色)。這個部分可以擴大圖像中的物體區域，填補小孔洞和斷裂。

```
def dilate(img, kernel):
    h, w = img.shape[:2]
    kh, kw = kernel.shape[:2]
    pad_h, pad_w = (kh - 1) // 2, (kw - 1) // 2
    padded_img = np.pad(img, ((pad_h, pad_h), (pad_w, pad_w)),
mode='constant')
    dilated_img = np.zeros_like(img)
    for i in range(h):
        for j in range(w):
            if np.any(padded_img[i:i+kh, j:j+kw] & kernel):
                dilated_img[i, j] = 255
    return dilated_img
```

這段程式碼實現了圖像的 Boundary Extraction，用於提取圖像的邊界輪廓。對灰度圖進行二值化處理後，再創建一個 3x3 的矩形結構元素，對二值圖進行腐蝕和膨脹操作，將膨脹結果減去腐蝕結果得到圖像的邊界。最後，對邊界圖進行反相操作，將邊界部分設為黑色，其餘部分設為白色。

```
image = cv2.imread('IMG_4.jpg')
gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image_RGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
# 二值化
img_binary = threshold(gray_img, 128, 255)
# 3*3 Structures Element
origin_el = getStructuringElement(cv2.MORPH_RECT,(3, 3))
# Erosion
erosion = erode(img_binary, origin_el)
# Dilation
dilation = dilate(img_binary, origin_el)
# Dilation - Erosion = boundary
boundary = dilation - erosion

# 二值圖畫素取反
result = 255 - boundary
```

這段程式碼實現了圖像的 Region Filling。讀取圖像並轉換為灰度圖和二值圖後，構造一個初始的 marker 圖像，在邊界處設置為 255，再利用原圖的補集得到 mask 圖像，限制填充區域。接著使用膨脹操作不斷擴大 marker，直到收斂為止，在每次膨脹後，將結果與 mask 取最小值，以控制填充範圍。最後將填充結果與原始二值圖相減，得到填充的區域。

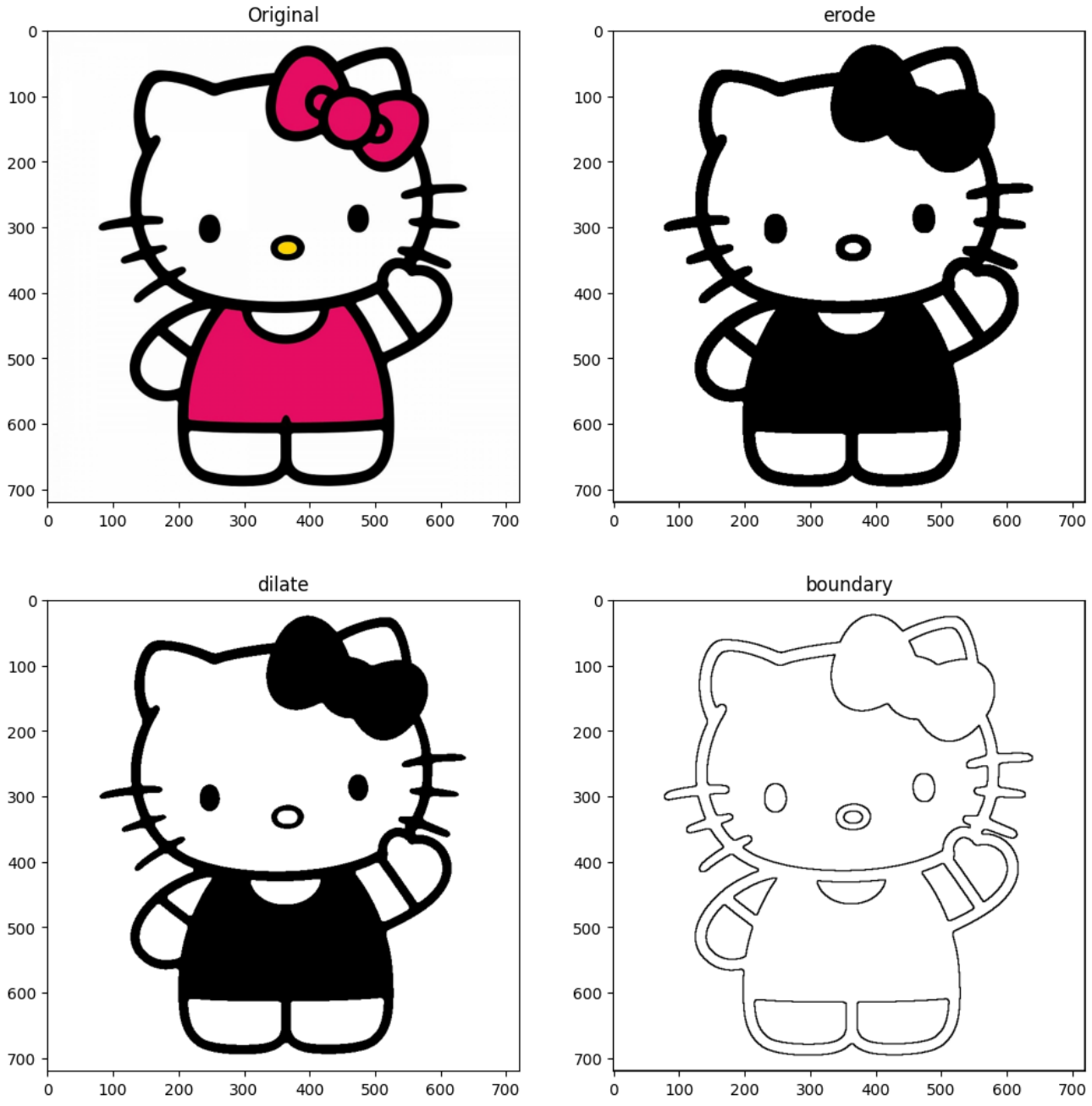
```
# 讀取圖像
image = cv2.imread('IMG_5.jpg')
image_RGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# 二值化
img_binary = threshold(gray_img, 128, 255)

# 構造 Marker 圖像
marker = np.zeros_like(img_binary)
marker[0, :] = 255
marker[-1, :] = 255
marker[:, 0] = 255
marker[:, -1] = 255

# 原圖取補得到 MASK 圖像,限制膨脹結果
mask = 255 - img_binary
# cv2_imshow(mask)

# Region Filling,定義一個結構元素做膨脹,直到收斂為止
origin_el = getStructuringElement(cv2.MORPH_CROSS,(3, 3))

while True:
    marker_pre = marker.copy()
    dilation = dilate(marker, origin_el)
    marker = np.minimum(dilation, mask)  # dilation 跟 mask 做比較，取最小值(0)，
以達到 filling 的效果
    if np.array_equal(marker_pre, marker):
        break

dst = 255 - marker
filling = dst - img_binary
```

Boundary Extraction Result:

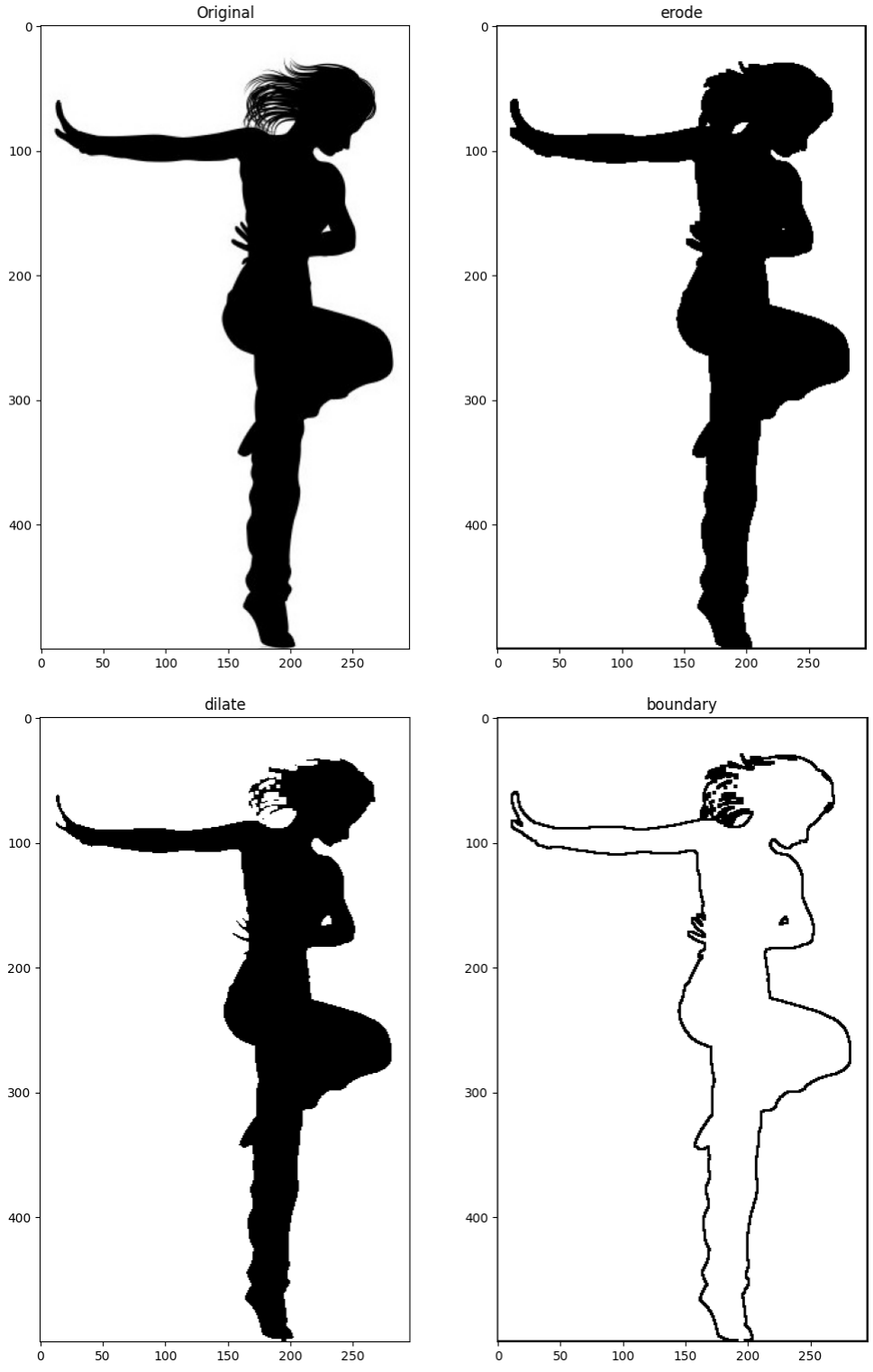
圖組一、

左上為原圖(input)，右上為經過 Erode 的圖，

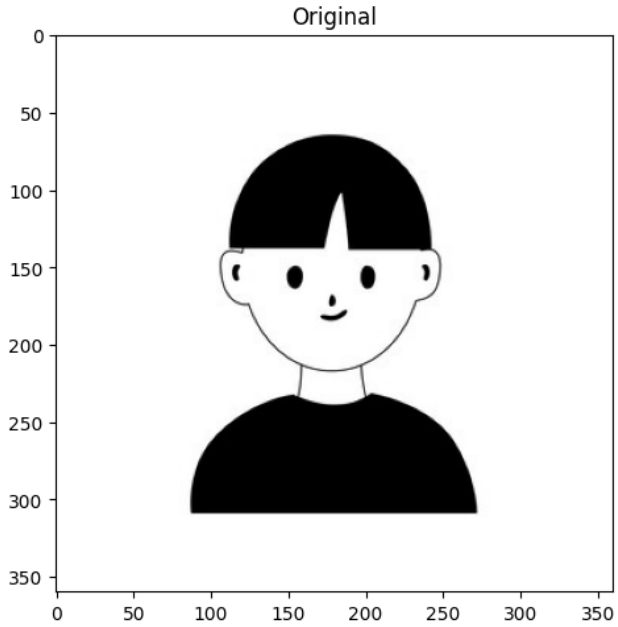
左下為經過 Dilate 的圖，右下為最後萃取出的邊界成果。

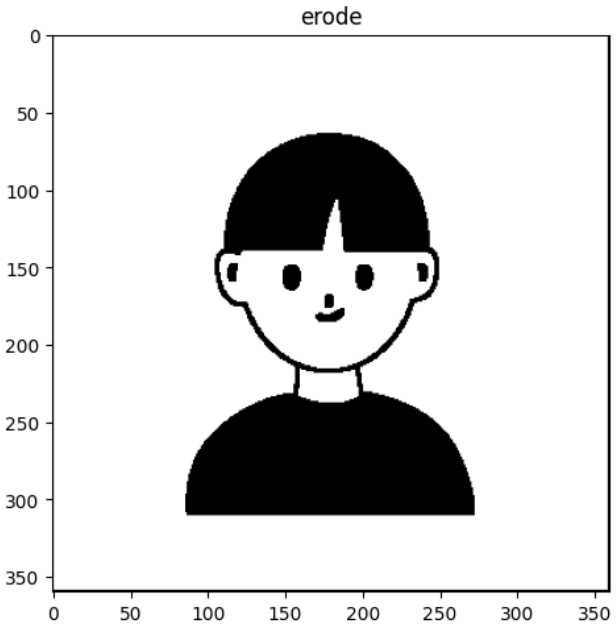


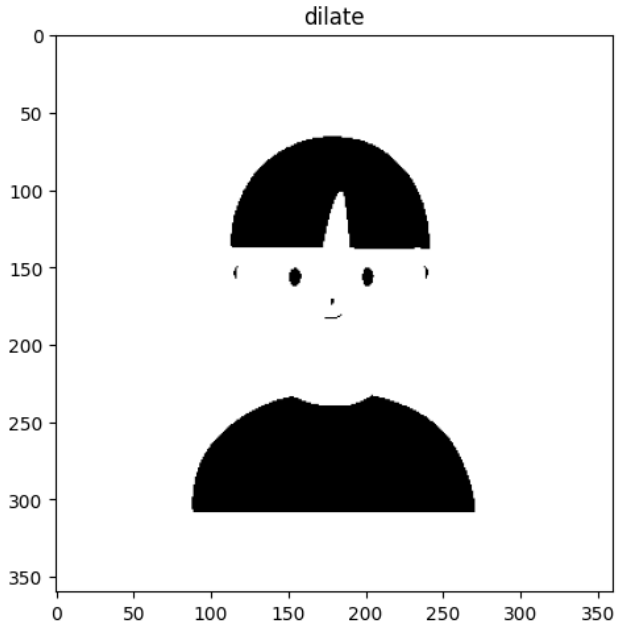
圖組二、

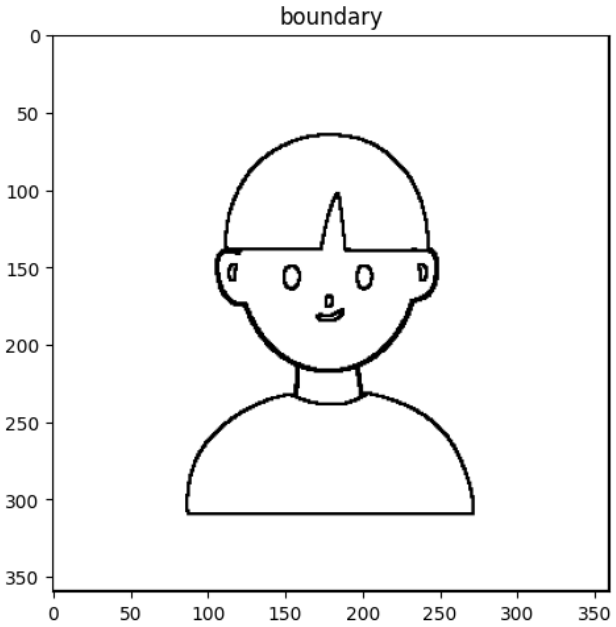


圖組三、

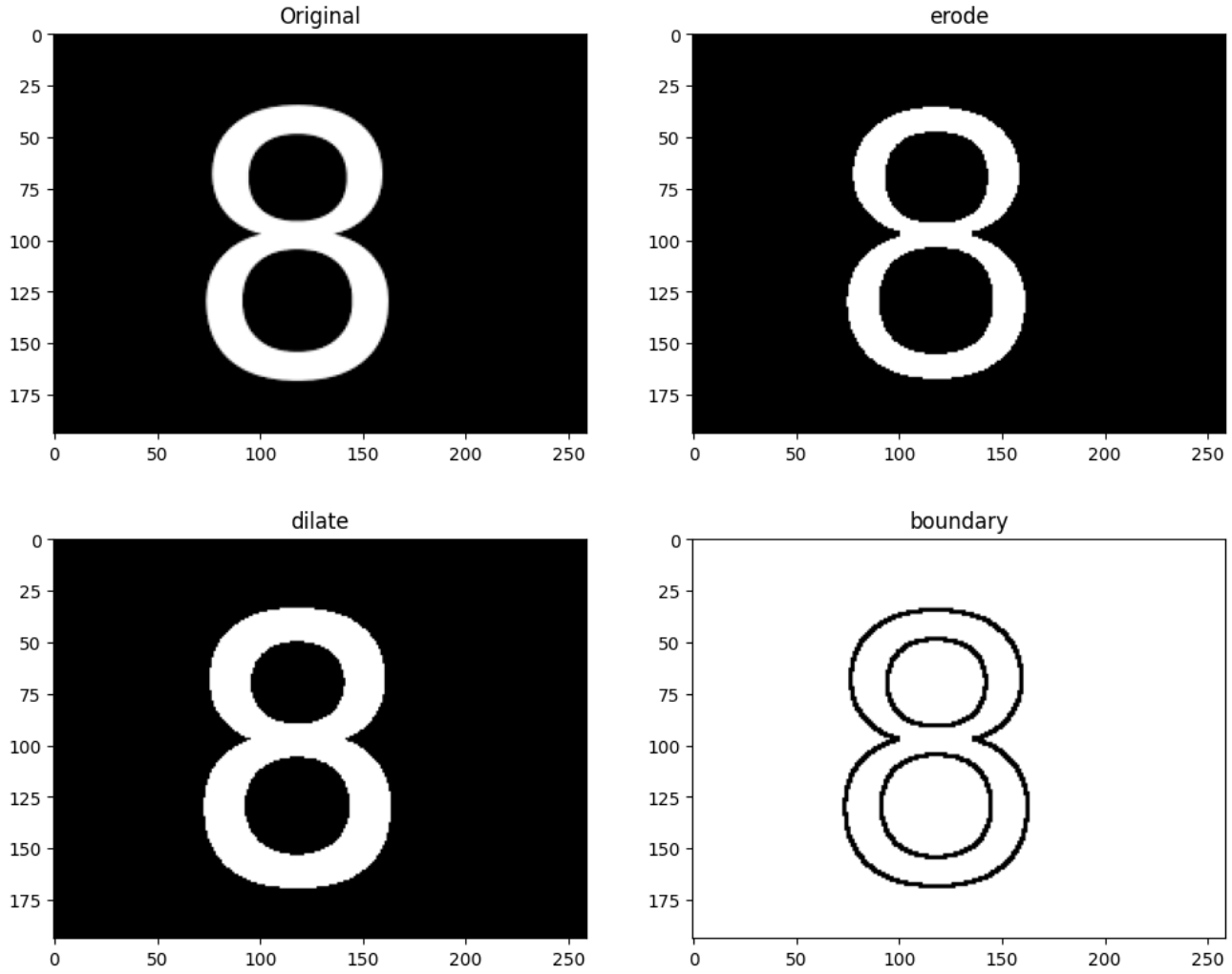

Original
0
50
100
150
200
250
300
350


erode
0
50
100
150
200
250
300
350


dilate
0
50
100
150
200
250
300
350


boundary
0
50
100
150
200
250
300
350

圖組四、

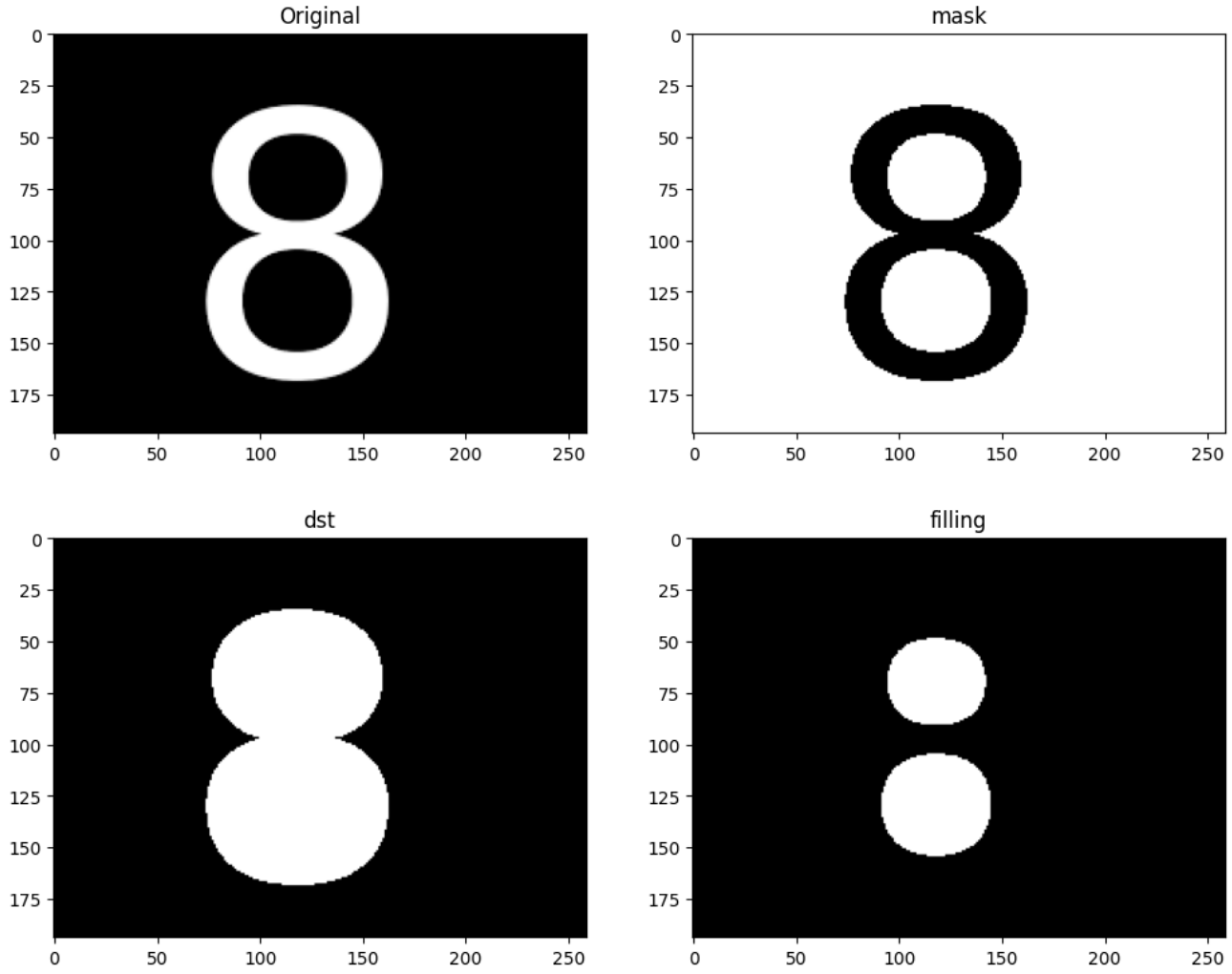

Original
erode
dilate
boundary
0
25
50
75
100
125
150
175
0
50
100
150
200
250

Region Filling Result:

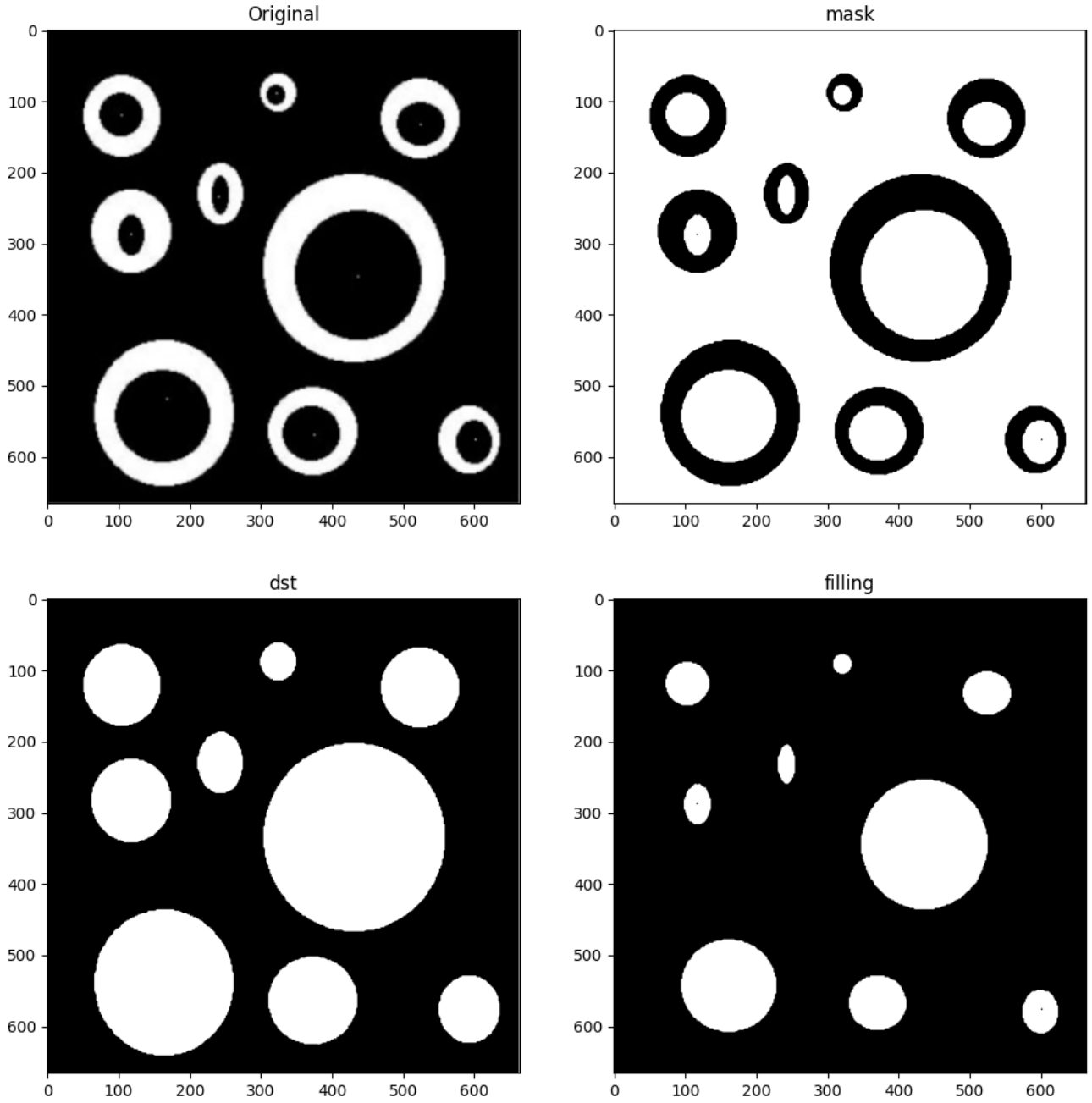
圖組一、

左上為原圖(input)，右上為經過 Mask 的圖限制區域填充的範圍，

左下表示經過區域填充後的圖，右下為填充區域與原始物體之間的差異。



圖組二、



Discussion:

1. 報告中實現了兩種基本的 Morphological Operations : Erosion(腐蝕) 和 Dilation(膨脹)。Erosion 可以消除圖像中的小雜點和毛刺，使圖像更加平滑，而 Dilation 可以擴大圖像中的物體區域，填補小孔洞和斷裂。利用 Morphological Operations 成功在邊界提取保留了物體的輪廓訊息，而區域填充則消除了物體內部的空心。
2. 在處理最後一張圖像時，使用 OpenCV 提供的內建函數 cv2.dilate()比自己實作的 dilate 函數運行得快速很多，也應證 OpenCV 庫經過了多年的發展和優化，在算法實現、低級語言編寫、硬體加速、向量化操作和內存管理等方面進行了大量的優化，使得其內建函數在運行速度上通常比自己實作的函數更有優勢。