HW4 report

R10922082 林育駿

本次作業中使用 python opencv 幫助圖片的 I/O 一些處理部分，執行 R10922082_HW4.py 會生成 dilation_lena.bmp, erosion_lena.bmp, opening_lena.bmp, closing_lena.bmp 和 hit_and_miss_lena.bmp 五張圖片對應五個小題。

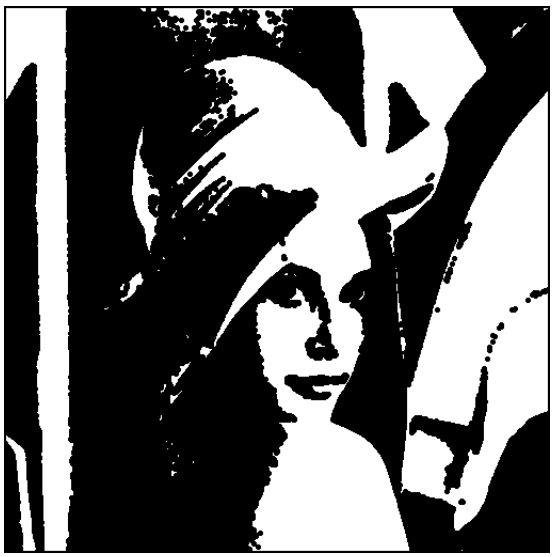
(a) Dilation

Lena 原圖做完 binarized 後利用指定的 3-5-5-5-3 kernel 做 dilation，在原圖上有值的地方將 kernel 原點對上去以 kernel 上為 1 的部分擴散出去，最後得到的圖可以發現一些小縫隙能被填補起來，超出邊界的部分則忽略不影響。

```
def dilation(img, kernel, k_h, k_w, centerX, centerY):      # if center has
    # print("do dilation")
    (imgHeight, imgWidth) = img.shape[:2]

    # init a white canvas
    nimg = np.zeros(shape=(imgWidth,imgHeight,3))
    for i in range(imgHeight):
        for j in range(imgWidth):
            if img[i][j].mean() == 255:
                for a in range(k_h):
                    for b in range(k_w):
                        if kernel[a][b] ==1 and i+a-centerY < imgHeight and
                            nimg[i+a-centerY][j+b-centerX] = img[i][j]
```

(b) Erosion



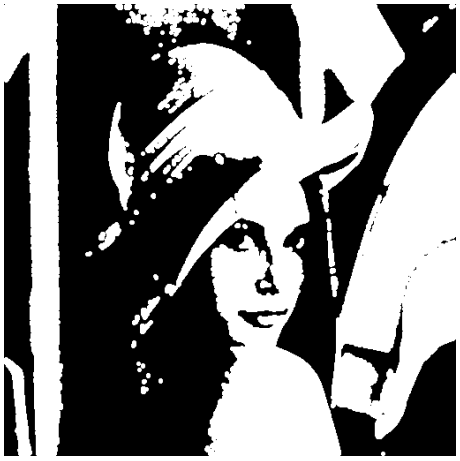
Lena 原圖做完 binarized 後利用指定的 3-5-5-5-3 kernel 做 erosion，在原圖上每個點將 kernel 原點對上去以 kernel 為想要尋找的 pattern 去看說是否有相同 pattern 出現在該點，最後得到的圖可以發現一些凸出去的部分或粗糙的邊界變得圓滑，超出邊界的部分則為 0。

```
def erosion(img,kernel, k_h,k_w, centerX,centerY):       # if the kernel match,
    # print("do erosion")
    (imgHeight, imgWidth) = img.shape[:2]
    # represent kernel information center point and half width and height

    # init a white canvas
    nimg = np.zeros(shape=(imgWidth,imgHeight,3))
    for i in range(imgHeight):
        for j in range(imgWidth):
            write = 1
            for a in range(k_h):  # kernel operation
                for b in range(k_w):
                    if kernel[a][b] ==1:
                        if i+a-centerY >= imgHeight or i+a-centerY < 0 or j+b-c
                            write =0

            if write ==1:
                nimg[i][j] =  np.array([255,255,255])
```

(c) Opening



Opening 為先 erosion 再 dilation，能發現圖上一些 component 分了開來且圓滑。

```
def openOp(img):        # erosion then dilation
    # print("do open")
    kernel =[[0,1,1,1,0],
             [1,1,1,1,1],
             [1,1,1,1,1],
             [1,1,1,1,1],
             [0,1,1,1,0]]
    e_img = erosion(img,kernel,5,5,2,2)
    o_img = dilation(e_img,kernel,5,5,2,2,)
    cv2.imwrite("opening_lena.bmp", o_img) #write the output of image
```

(d) Closing

Closing 為先 dilation 再 erosion，和 opening 的操作順序正好相反。

```python
def openOp(img):        # erosion then dilation
    # print("do open")
    kernel =[[0,1,1,1,0],
             [1,1,1,1,1],
             [1,1,1,1,1],
             [1,1,1,1,1],
             [0,1,1,1,0]]
    e_img = erosion(img,kernel,5,5,2,2)
    o_img = dilation(e_img,kernel,5,5,2,2,)
    cv2.imwrite("opening_lena.bmp", o_img) #write the output of image
```

(e) Hit-and-miss transform

Hit and miss 後的 lena 圖可以發現找出了指定 pattern 在圖上出現的位置，利用兩個稍微位移的 kernel 分別對 binarized 的 lena 圖和其負片做 erosion，兩張的結果可以抓出指定 pattern 出現的位置，再利用兩張圖做交集得到真正 kernel pattern 在圖上出現的邊界，例如此題即找出 L 型右上角。

```python
hit_and_miss(img, j_kernel, k_kernel, j_x,j_y,k_x,k_y,k_h,k_w):
(imgHeight, imgWidth) = img.shape[:2]

rimg = np.zeros(shape=(imgWidth,imgHeight,3))
for i in range(imgHeight):
    for j in range(imgWidth):
        if img[i][j].mean() ==255:
            rimg[i][j] = np.array([0,0,0])
        if img[i][j].mean() ==0:
            rimg[i][j] = np.array([255,255,255])
rimg = rimg.astype(np.uint8) #set the right data type

# get two results of erosion image with different kernel
je_img = erosion(img,j_kernel,k_h,k_w,j_x,j_y)
ke_img = erosion(rimg,k_kernel,k_h,k_w,k_x,k_y)

fimg = np.zeros(shape=(imgWidth,imgHeight,3))
for i in range(imgHeight):
    for j in range(imgWidth):
        if je_img[i][j].mean() == 255 and ke_img[i][j].mean() ==255:
            fimg[i][j] = np.array([255,255,255])

fimg = fimg.astype(np.uint8) #set the right data type
cv2.imwrite("hit_and_miss_lena.bmp", fimg) #write the output of image
```