Computer Vision HW4 楊閔喻 R09921012

Write a program to generate images and histograms:

(a) Dilation
(b) Erosion
(c) Opening
(d) Closing
(e) Hit-and-miss transform

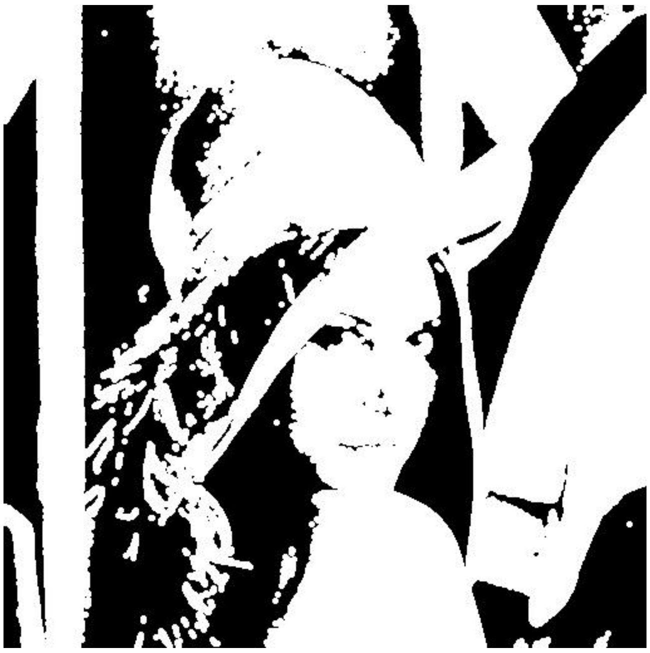
(a)

```
def Dilation(image_output,image,ker,ker_num):
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if image[i,j] == 255:
                for k in range(kernal_num):
                    if i+ker[k,0]>=0 and i+ker[k,0]<image.shape[0] and
j+ker[k,1]>=0 and j+ker[k,1]<image.shape[1]:
                        image_output[i+ker[k,0],j+ker[k,1]] = 255
```

```
kernal[0,0],kernal[0,1] = -2,-1
kernal[1,0],kernal[1,1] = -2,0
kernal[2,0],kernal[2,1] = -2,1
kernal[3,0],kernal[3,1] = -1,-2
kernal[4,0],kernal[4,1] = -1,-1
kernal[5,0],kernal[5,1] = -1,0
kernal[6,0],kernal[6,1] = -1,1
kernal[7,0],kernal[7,1] = -1,2
kernal[8,0],kernal[8,1] = 0,-2
kernal[9,0],kernal[9,1] = 0,-1
kernal[10,0],kernal[10,1] = 0,0
kernal[11,0],kernal[11,1] = 0,1
kernal[12,0],kernal[12,1] = 0,2
kernal[13,0],kernal[13,1] = 1,-2
kernal[14,0],kernal[14,1] = 1,-1
kernal[15,0],kernal[15,1] = 1,0
```

```
kernal[16,0],kernal[16,1] = 1,1
kernal[17,0],kernal[17,1] = 1,2
kernal[18,0],kernal[18,1] = 2,-1
kernal[19,0],kernal[19,1] = 2,0
kernal[20,0],kernal[20,1] = 2,1
```

- 在主程式中使用二維陣列紀錄 kernal
- 在 Dilation 函式中，對每個 pixel 的 kernal 範圍進行處裡，使其 intensity 為 255

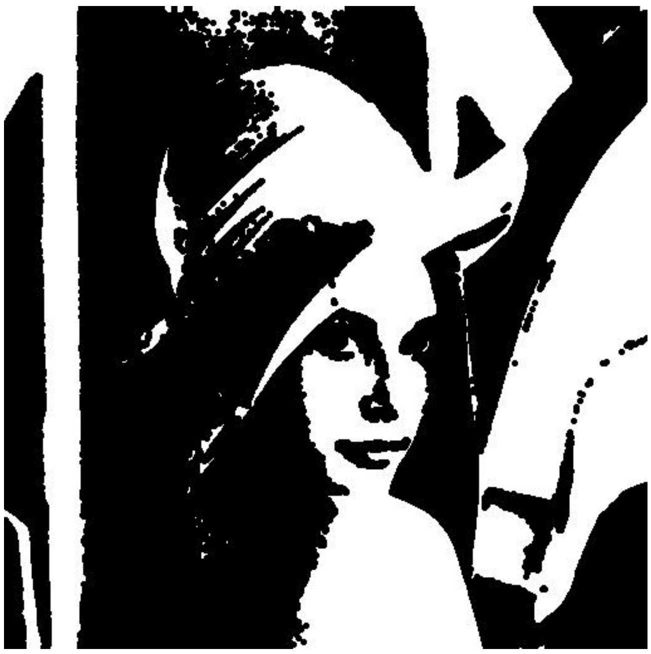


(b)

```
def Erosion(image_output,image,ker,ker_num):
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            temp = True
            for k in range(ker_num):
                if i+ker[k,0]>=0 and i+ker[k,0]<image.shape[0] and j+ke
r[k,1]>=0 and j+ker[k,1]<image.shape[1] and image[i+ker[k,0],j+ker[k,1]
] == 0:
                    temp = False
                    break
            if temp == True:
                image_output[i,j] = 255
```

- 在 Erosion 函式中，對於給個 pixel 點的 kernal 範圍進行判斷，如果 kernal

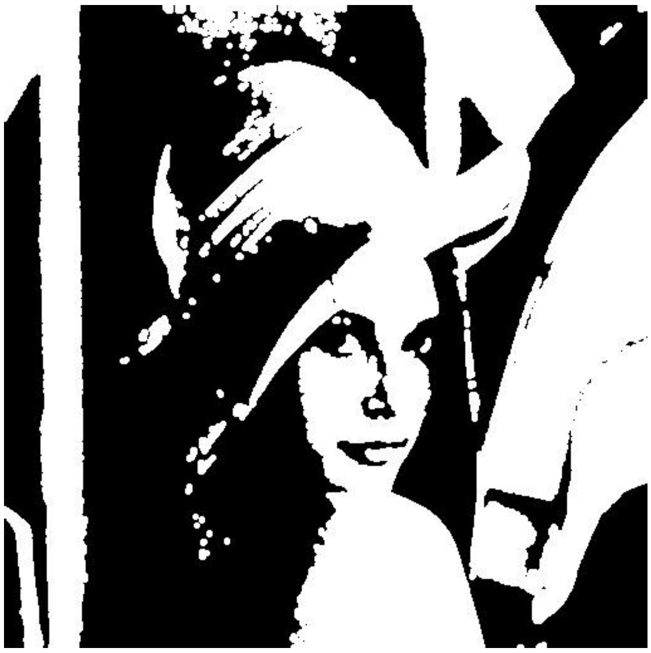
範圍內有點的 intensity 值不為 255，則設定該 pixel 之 instensity 為 0



(c)、(d)

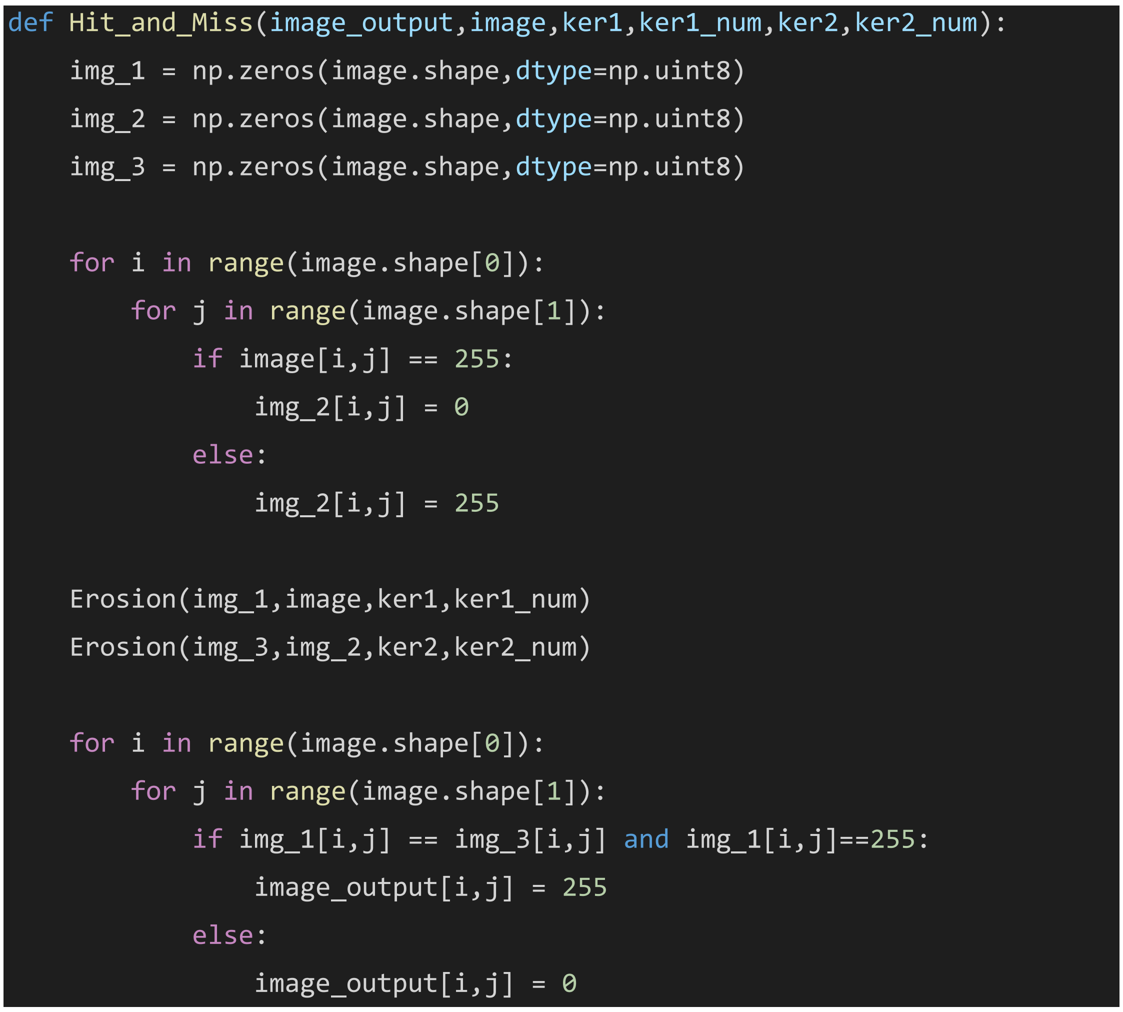
```
print("Start Opening")
Dilation(img_c,img_b,kernal,kernal_num)
print("Start Closing")
Erosion(img_d,img_a,kernal,kernal_num)
```

- Opening 為在主程式中，對已經過 Erosion 的原圖再進行 Dilation



- Closing 為在主程式中，對已經過 Dilation 的原圖再進行 Erosion

(e)



```
def Hit_and_Miss(image_output,image,ker1,ker1_num,ker2,ker2_num):
    img_1 = np.zeros(image.shape,dtype=np.uint8)
    img_2 = np.zeros(image.shape,dtype=np.uint8)
    img_3 = np.zeros(image.shape,dtype=np.uint8)

    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if image[i,j] == 255:
                img_2[i,j] = 0
            else:
                img_2[i,j] = 255

    Erosion(img_1,image,ker1,ker1_num)
    Erosion(img_3,img_2,ker2,ker2_num)

    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            if img_1[i,j] == img_3[i,j] and img_1[i,j]==255:
                image_output[i,j] = 255
            else:
                image_output[i,j] = 0
```

```
kernal_l[0,0],kernal_l[0,1] = 0,0
kernal_l[1,0],kernal_l[1,1] = 0,-1
kernal_l[2,0],kernal_l[2,1] = 1,0
kernal_l2[0,0],kernal_l2[0,1] = -1,0
kernal_l2[1,0],kernal_l2[1,1] = -1,1
kernal_l2[2,0],kernal_l2[2,1] = 0,1
```

- Hit-and-Miss 函式中，先對原圖取補集，而後將原圖與其補集分別使用兩個 L 型 kernal 做 Erosion，並在最後取兩者 Erosion 結果的交集作為輸出