Cv2 hw5

R10521516 廖浚評

Lena 原圖

Use python opencv 讀入圖片

(1)

dilation(如果捲積 kernel 範圍內所有像素值都存在(>0)，那麼新的像素值就選擇 kernel 內的最大值，這表示捲積 kernel 掃過的所有像素都會被膨脹掉(變為最大值)，所以整張影像的黑色區域會變少。

```
def dilation(img_in , kernel):
    shape = img_in.shape
    img_dil = np.zeros(img_in.shape , dtype = int)
    for i in range(shape[0]):
        for j in range(shape[1]):
            if img_in[i][j] > 0:
                maxv= 0 #設定成0
                for kernel_each in kernel:
                    kernel_i ,kernel_j = kernel_each
                    if (i+kernel_i)>=0 and(i+kernel_i)<=(shape[0]-1)and \
                        (j+kernel_j)>=0 and(j+kernel_j)<=(shape[1]-1):
                            if img_in[i+kernel_i,j+kernel_j]>maxv:
                                maxv = (img_in[i+kernel_i,j+kernel_j])
                for kernel_each in kernel:
                    kernel_i ,kernel_j = kernel_each
                    if (i+kernel_i)>=0 and(i+kernel_i)<=(shape[0]-1)and \
                        (j+kernel_j)>=0 and(j+kernel_j)<=(shape[1]-1):
                            img_dil[i+kernel_i,j+kernel_j]=maxv

    return img_dil
```

(2)erosion

如果捲積 kernel 範圍內所有像素值都存在(>0)，那麼新的像素值就選擇 kernel 內的最小值，這表示捲積 kernel 掃過的所有像素都會被腐蝕或侵蝕掉(變為最小值)，所以整張影像的白色區域會變少。

```
def erosion(img_in , kernel):
    shape = img_in.shape
    img_ero = np.zeros(img_in.shape , dtype = int)
    for i in range(shape[0]):
        for j in range(shape[1]):

                img_ero[i, j] = 255
                minv = np.inf #設定成無限大
                for kernel_each in kernel:
                    kernel_i ,kernel_j = kernel_each
                    if (i+kernel_i)>=0 and(i+kernel_i)<shape[0]and \
                        (j+kernel_j)>=0 and(j+kernel_j)<shape[1]:
                            if img_in[i+kernel_i,j+kernel_j]<minv:
                                minv =(img_in[i+kernel_i,j+kernel_j])
                for kernel_each in kernel:
                    kernel_i ,kernel_j = kernel_each
                    if (i+kernel_i)>=0 and(i+kernel_i)<shape[0]and \
                        (j+kernel_j)>=0 and(j+kernel_j)<shape[1]:
                        img_ero[i+kernel_i,j+kernel_j] =minv
    return img_ero
```

(3) opening

```
def opening(img_in ,kernel ):
    img_open = erosion(dilation(img_in , kernel), kernel)
    return img_open
```

(4) closing

```
def closing(img_in , kernel):
    img_close = dilation(erosion(img_in , kernel), kernel)
    return img_close
```