Computer Vision HW4

R08922a27 資工系 人工智慧碩士班 李吉昌

I use python 3.7 to implement all image processing requirements. Reading .bmp file by **PIL**, and then processing through **NumPy** array.

- ## (a) Dilation

  1. ### Results

  2. ### Code fragment

```
def dilation(bin_img, kernel):
    img_dil = np.zeros(bin_img.shape).astype(int)
    for i in range(bin_img.shape[0]):
        for j in range(bin_img.shape[1]):
            if bin_img[i, j] > 0:
                for (p, q) in kernel:
                    i_dil, j_dil = i + p, j + q
                    if i_dil >= 0 and j_dil >= 0 and \
```

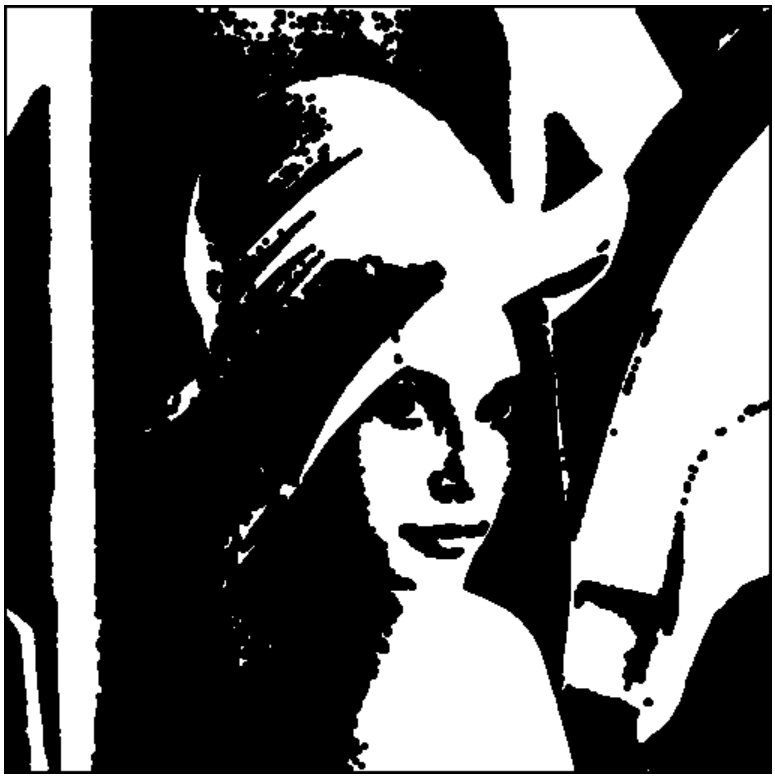
```
                            i_dil <= (bin_img.shape[0] - 1) and j_dil <=
(bin_img.shape[1] - 1):
                        img_dil[i_dil, j_dil] = 255
    return img_dil
img_dil = dilation(bin_img, kernel)
PIL_image = Image.fromarray(img_dil.astype('uint8'))
PIL_image.save('results/Dilation.bmp')
```

3. Brief description

The dilation function is defined. Those pixels in which neighbors in the range of kernel are not 0 would be assigned to 255.

• (b) Erosion

1. Results



2. Code fragment

```
def erosion(bin_img, kernel):
    img_ero = np.zeros(bin_img.shape).astype(int)
    for i in range(bin_img.shape[0]):
        for j in range(bin_img.shape[1]):
            Isdraw = True
            for (p, q) in kernel:
                i_dil, j_dil = i + p, j + q
```

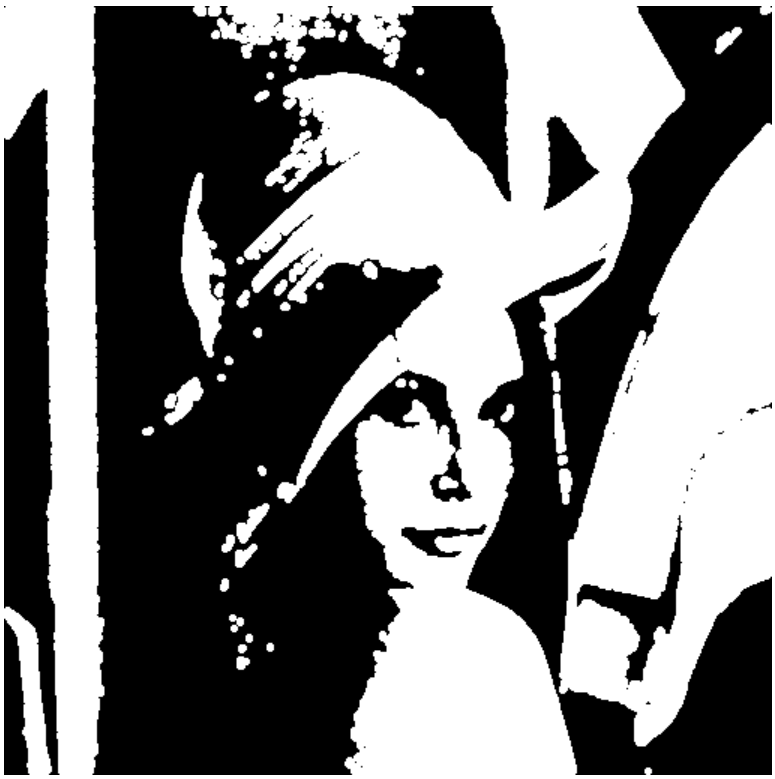
```
                if not(i_dil >= 0 and j_dil >= 0 and
                       i_dil <= (
                           bin_img.shape[0] - 1) and j_dil <=
(bin_img.shape[1] - 1)
                       and bin_img[i_dil, j_dil] > 0):
                    Isdraw = False
                    break
            if Isdraw:
                img_ero[i, j] = 255
    return img_ero
img_ero = erosion(bin_img, kernel)
PIL_image = Image.fromarray(img_ero.astype('uint8'))
PIL_image.save('results/Erosion.bmp')
```

3. **Brief description**

   The erosion function is defined. Those pixels in which neighbors in the range of kernel are all 255 would be assigned to 255.

- ## (c) Opening

1. **Results**



2. **Code fragment**

```
img_opn = dilation(erosion(bin_img, kernel), kernel)
PIL_image = Image.fromarray(img_opn.astype('uint8'))
PIL_image.save('results/Opening.bmp')
```

3. **Brief description**

   The previously defined function, dilation, and erosion are used. Starting with the erosion, and then the dilation is conducted.

- # (d) Closing

1. **Results**

2. **Code fragment**

```
img_cls = erosion(dilation(bin_img, kernel), kernel)
PIL_image = Image.fromarray(img_cls.astype('uint8'))
PIL_image.save('results/Closing.bmp')
```

3. **Brief description**

   The previously defined function, dilation, and erosion are used. Starting with the dilation, and then the erosion is conducted.

- ## (e) Hit-and-miss transform

1. Results

2. Code fragment

```
J_kernel = [[0, -1], [0, 0], [1, 0]]
K_kernel = [[-1, 0], [-1, 1], [0, 1]]
def hit_and_miss(bin_img, J_kernel, K_kernel):
img_ham = np.ones(bin_img.shape).astype(int) * 255
img_ham[np.logical_or(erosion(bin_img, J_kernel) < 128,
erosion(-bin_img + 255, K_kernel) < 128)] = 0
return img_ham
img_ham = hit_and_miss(bin_img, J_kernel, K_kernel)
PIL_image = Image.fromarray(img_ham.astype('uint8'))
PIL_image.save('results/HitAndMiss.bmp')
```

3. Brief description

The original binary image and the component one are conducted the erosion with J kernel and K kernel, respectively. Those pixels which are both 255 in the above results would be assigned to 255.