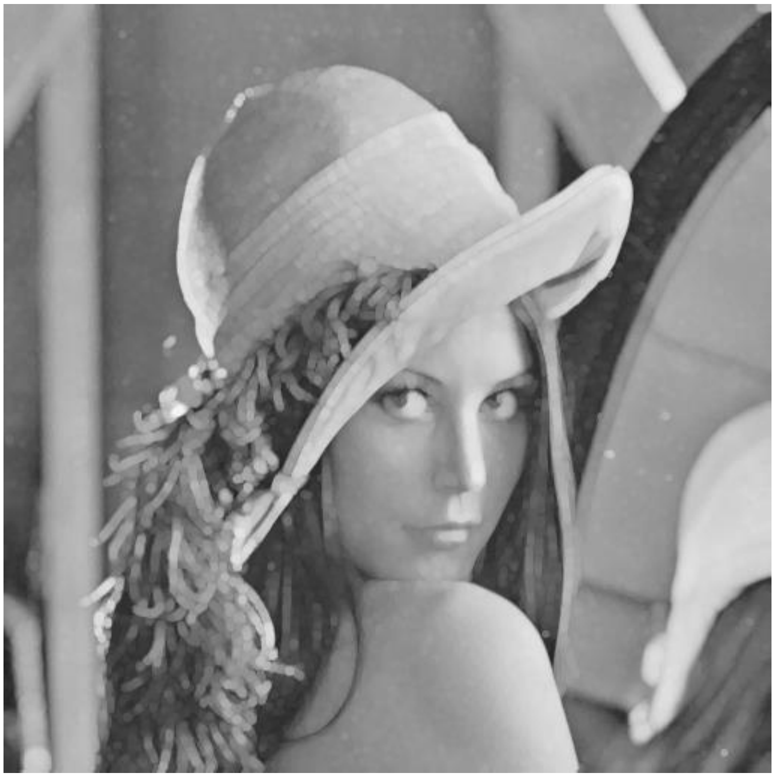
Computer Vision HW5

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

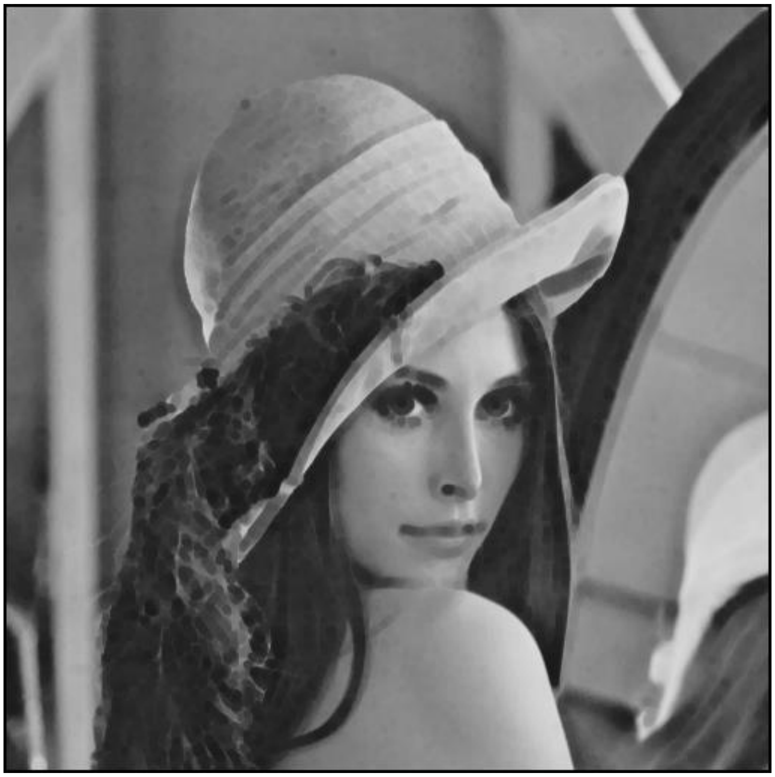
```
def dilation(sample_arr, kernel):
    img_dil = np.zeros(sample_arr.shape).astype(int)
    for i in range(sample_arr.shape[0]):
        for j in range(sample_arr.shape[1]):
            if sample_arr[i, j] > 0:
                max_val = 0
                for (p, q) in kernel:
                    i_dil,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

                for (p, q) in kernel:
                    i_dil, j_dil = i + p, j + q
                    if i_dil >= 0 and j_dil >= 0 and \
                            i_dil <= (sample_arr.shape[0] - 1) and j_dil
<= (sample_arr.shape[1] - 1):
                        img_dil[i_dil, j_dil] = max_val
    return img_dil
img_dil = dilation(sample_arr, kernel)
PIL_image = Image.fromarray(img_dil.astype('uint8'))
PIL_image.save('results/Dilation.bmp')
```

3. Brief description

The dilation function is defined. Those grey-level values of pixels in which neighbors in the kernel range are not 0 would be assigned to the maximum grey value of those pixels in the kernel field.

- ## (b) Erosion

1. Results



2. Code fragment

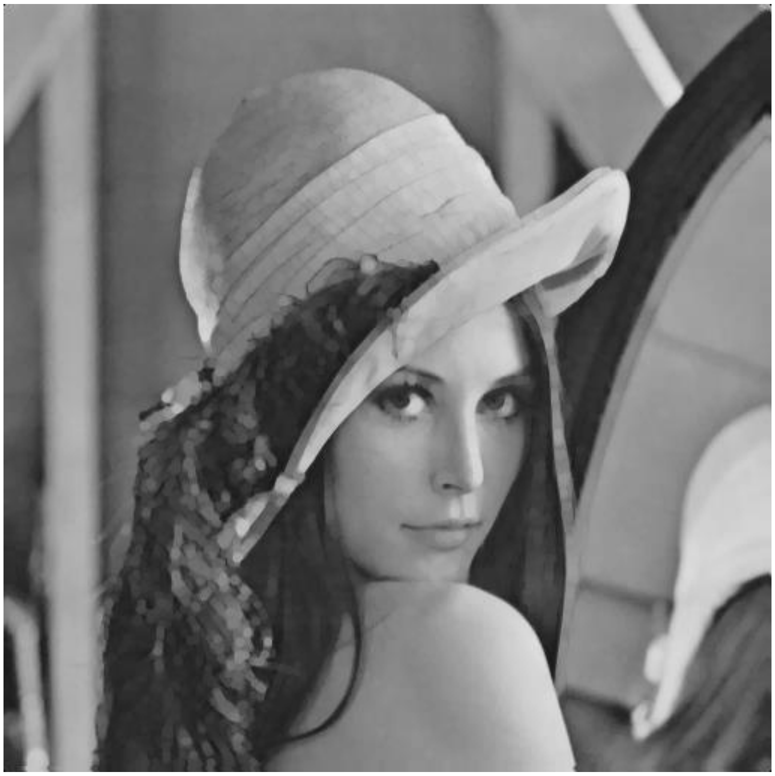
```
def erosion(sample_arr, kernel):
    img_ero = np.zeros(sample_arr.shape).astype(int)
    for i in range(sample_arr.shape[0]):
        for j in range(sample_arr.shape[1]):
            Isdraw = True
            min_val = 255
            for (p, q) in kernel:
                i_dil, j_dil = i + p, j + q
                if not(i_dil >= 0 and j_dil >= 0 and
                       i_dil <= (
                           sample_arr.shape[0] - 1) and j_dil <=
(sample_arr.shape[1] - 1)
                       and sample_arr[i_dil, j_dil] > 0):
                    Isdraw = False
                    break
                min_val = min(min_val, sample_arr[i_dil, j_dil])
            if Isdraw:
                for (p, q) in kernel:
                    i_dil, j_dil = i + p, j + q
                    if i_dil >= 0 and j_dil >= 0 and i_dil <= (
                            sample_arr.shape[0] - 1) and j_dil <=
(sample_arr.shape[1] - 1) \
                            and sample_arr[i_dil, j_dil] > 0:
                        img_ero[i, j] = min_val
    return img_ero
img_ero = erosion(sample_arr, kernel)
PIL_image = Image.fromarray(img_ero.astype('uint8'))
PIL_image.save('results/Erosion.bmp')
```

3. Brief description

The erosion function is defined. Those grey level values of pixels in which neighbors in the kernel range are all 255 would be assigned to the minimum grey value of those pixels in the kernel field.

- # (c) Opening

1. Results



2. Code fragment

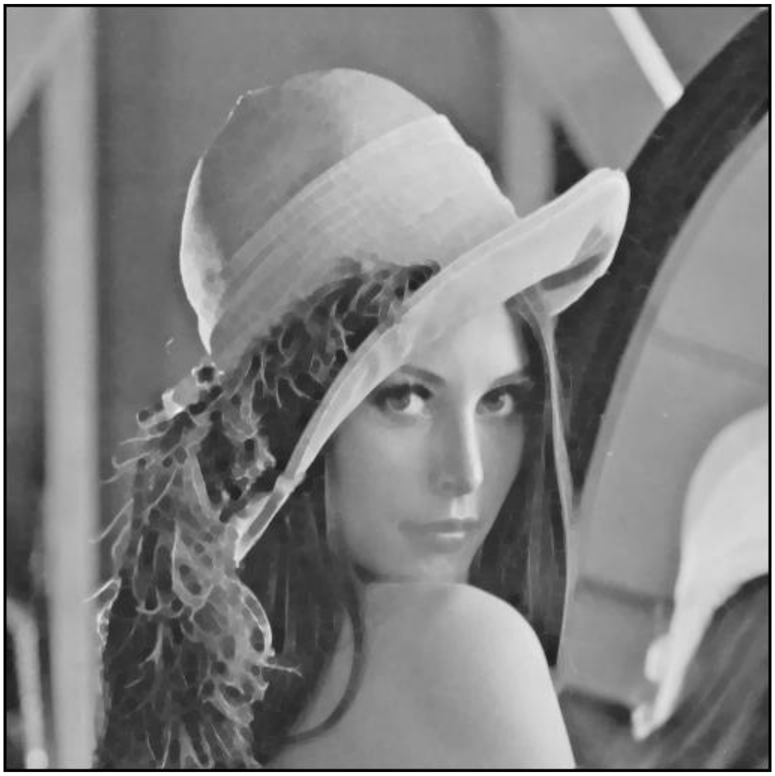
```
img_opn = dilation(erosion(sample_arr, kernel), kernel)
PIL_image = Image.fromarray(img_opn.astype('uint8'))
PIL_image.save('results/Opening.bmp')
```

3. Brief description

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

- ## (d) Closing

1. Results



2. Code fragment

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

3. Brief description

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