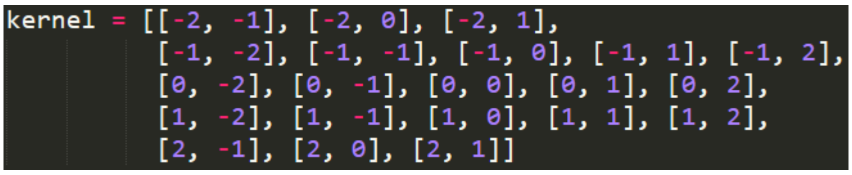
B06502149 資工三 張琦琛

(a) Dilation on binary image

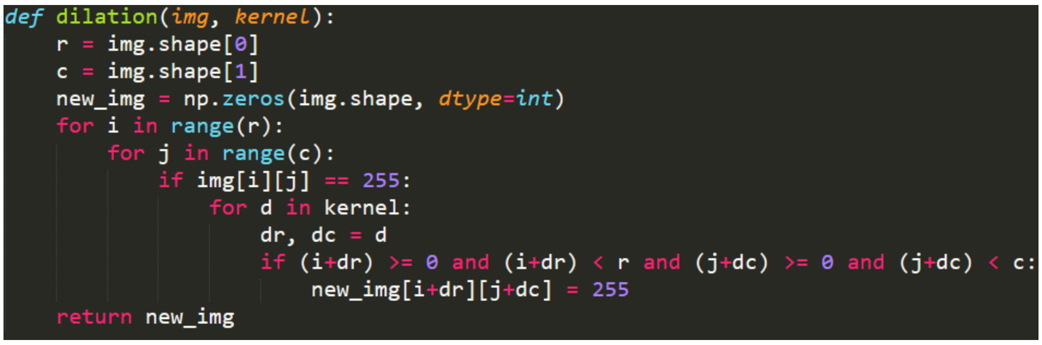
Algorithm:

$$A \oplus B = \{c \in E^N \mid c = a + b \text{ for some } a \in A \text{ and } b \in B\}$$

Code fragment:

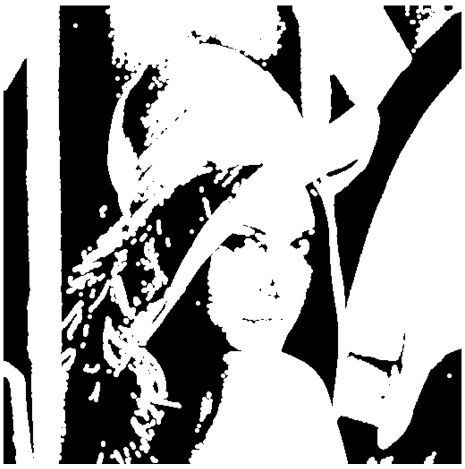


```
kernel = [[-2, -1], [-2, 0], [-2, 1],
          [-1, -2], [-1, -1], [-1, 0], [-1, 1], [-1, 2],
          [0, -2], [0, -1], [0, 0], [0, 1], [0, 2],
          [1, -2], [1, -1], [1, 0], [1, 1], [1, 2],
          [2, -1], [2, 0], [2, 1]]
```



```
def dilation(img, kernel):
    r = img.shape[0]
    c = img.shape[1]
    new_img = np.zeros(img.shape, dtype=int)
    for i in range(r):
        for j in range(c):
            if img[i][j] == 255:
                for d in kernel:
                    dr, dc = d
                    if (i+dr) >= 0 and (i+dr) < r and (j+dc) >= 0 and (j+dc) < c:
                        new_img[i+dr][j+dc] = 255
    return new_img
```

Result:

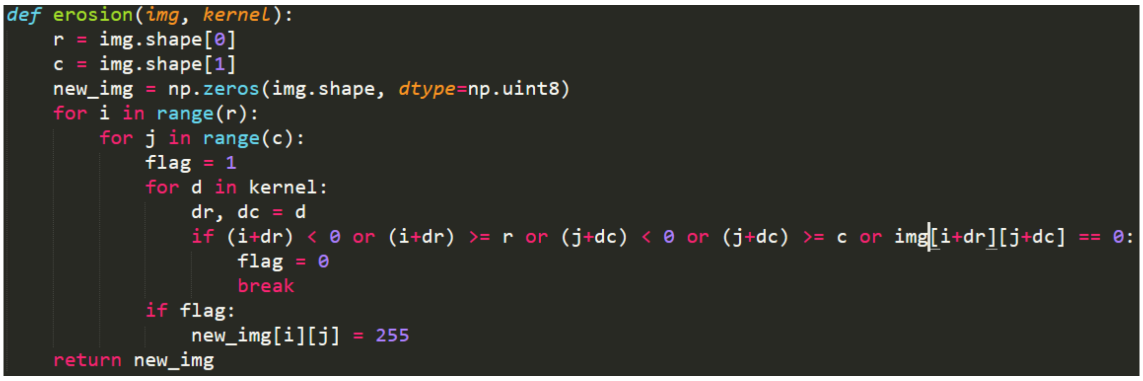


(b) Erosion on binary image

Algorithm:

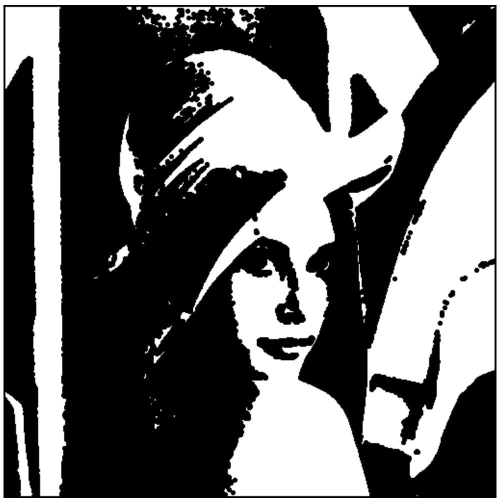
$$A \ominus B = \{x \in E^N \mid x + b \in A \text{ for every } b \in B\}$$

Code fragment:



```python
def erosion(img, kernel):
    r = img.shape[0]
    c = img.shape[1]
    new_img = np.zeros(img.shape, dtype=np.uint8)
    for i in range(r):
        for j in range(c):
            flag = 1
            for d in kernel:
                dr, dc = d
                if (i+dr) < 0 or (i+dr) >= r or (j+dc) < 0 or (j+dc) >= c or img[i+dr][j+dc] == 0:
                    flag = 0
                    break
            if flag:
                new_img[i][j] = 255
    return new_img
```

Result:

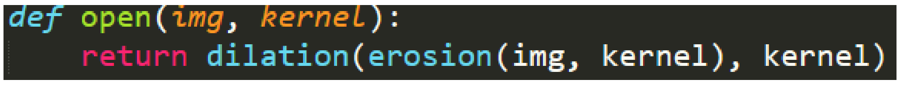


(c) Opening on binary image

Algorithm:

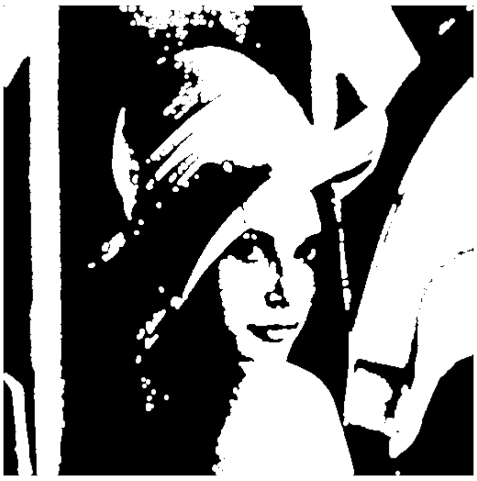
$$B \circ K = (B \ominus K) \oplus K$$

Code fragment:



```python
def open(img, kernel):
    return dilation(erosion(img, kernel), kernel)
```

Result:



(d) Closing on binary image

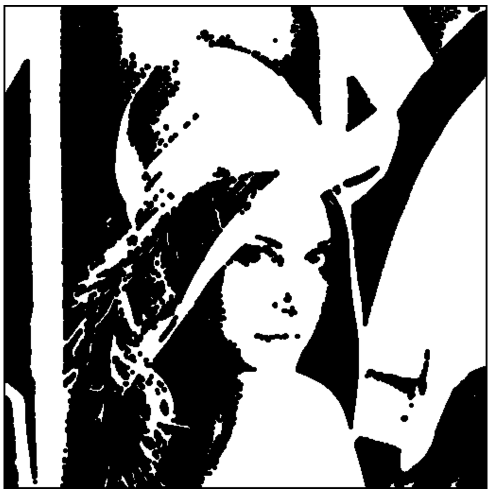
Algorithm:

$$B \bullet K = (B \oplus K) \ominus K$$

Code fragment:

```python
def close(img, kernel):
    return erosion(dilation(img, kernel), kernel)
```

Result:



(e) Hit and Miss on binary image

Algorithm:

$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$$

Code fragment:

```python
def hitAndMiss(img):
    J_kernel = [[0, -1], [0, 0], [1, 0]]
    K_kernel = [[-1, 0], [-1, 1], [0, 1]]
    img_comp = comp(img)
    r = img.shape[0]
    c = img.shape[1]
    new_img = np.zeros(img.shape, dtype=int)
    tmp1 = erosion(img, J_kernel)
    tmp2 = erosion(img_comp, K_kernel)
    for i in range(r):
        for j in range(c):
            if (tmp1[i][j] == 255 and tmp2[i][j] == 255):
                new_img[i][j] = 255
    return new_img
```

Result: