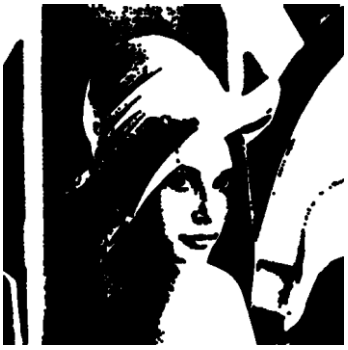
2021 CV HW4 Report

Results

a. dilation

b. erosion



c. opening

d. closing

e. hit and miss

(a) Dilation

Step1. 首先定義 octagonal 3-5-5-5-3 kernel。

```
kernel = np.array([[0, 1, 1, 1, 0],
                   [1, 1, 1, 1, 1],
                   [1, 1, 1, 1, 1],
                   [1, 1, 1, 1, 1],
                   [0, 1, 1, 1, 0]])
hh, ww = kernel.shape
```

Step2. 每個 pixel 作為 kernel 中心點，若此 pixel value 為 255，則將 kernel 為 1 的地方 pixel value 設為 255；若此 pixel value 不為 255，則不做事。

```
output = np.zeros((h, w), np.uint8)
for i in range(h):
    for j in range(w):
```

```
        if img[i][j] != 255:
            continue
        for ii in range(hh):
            for jj in range(ww):
                if kernel[ii][jj] == 1:
                    if i+ii-2 < 0 or j+jj-2 < 0 or i+ii-2 >= h or j+jj-2
>= w:
                        continue
                    output[i+ii-2][j+jj-2] = 255
```

(b) Erosion

每個 pixel 作為 kernel 中心點，若所有 kernel 為 1 的地方 pixel 都為 255，則此 pixel value 設為 255；反之，若 kernel 為 1 的地方有 pixel 不為 255，則此 pixel value 為 0。

```
output = np.zeros((h, w), np.uint8)
for i in range(h):
    for j in range(w):
        flag = 0
        for ii in range(hh):
            for jj in range(ww):
                if kernel[ii][jj] == 1:
                    if i+ii-2 < 0 or j+jj-2 < 0 or i+ii-2 >= h or j+jj-2
>= w:
                        continue
                    if img[i+ii-2][j+jj-2] != 255:
                        flag = 1
                        break
        if flag == 0:
            output[i][j] = 255
```

(c) Opening

承 a、b 的程式，先做 erosion 再做 dilation。

```
erosion_img = erosion(img)
opening_img = dilation(erosion_img)
```

(d) Closing

承 a、b 的程式，先做 dilation 再做 erosion。

```
dilation_img = dilation(img)
closing_img = erosion(dilation_img)
```

(e) Hit-and-miss transform

Step1. 首先定義兩個 kernel。

```
J = np.array([[0, 0, 0],
              [1, 1, 0],
              [0, 1, 0]])
K = np.array([[0, 1, 1],
              [0, 0, 1],
              [0, 0, 0]])
```

Step2. 用 kernel J 對原圖做 erosion 以及用 kernel K 對原圖的補集（255-原圖）做 erosion

```
img_J = erosion(img, J)
img_K = erosion(255-img, K)
```

Step3. 最後將兩張做完 erosion 的圖片取交集，即為 hit-and-miss 的結果。

```
output = np.zeros((h, w), np.uint8)
for i in range(h):
    for j in range(w):
        if img_J[i][j] == 255 and img_K[i][j] == 255:
            output[i][j] = 255
```