

## Description:

1. Binarize lena.bmp with threshold 128
2. Down sampling lena.bmp from 512x512 to 64x64, and using 8x8 blocks as a unit, take the topmost-left pixel as the downsampled data.
3. Count the Yokoi connectivity number on a downsampled lena using 4-connected.

## Algorithm:

Yokoi h function for 4-connectivity

$$h(b, c, d, e) = \begin{cases} q & \text{if } b = c \text{ and } (d \neq b \vee e \neq b) \\ r & \text{if } b = c \text{ and } (d = b \wedge e = b) \\ s & \text{if } b \neq c \text{ and } (d = b \wedge e = b) \end{cases}$$

Yokoi f function for 4-connectivity

$$f(a_1, a_2, a_3, a_4) = \begin{cases} 5, & \text{if } a_1 = a_2 = a_3 = a_4 = r \\ n, & \text{where } n = \text{number of } \{a_k | a_k = q\}, \text{ otherwise} \end{cases}$$

## Code fragment:

```
def Yokoi(img):
    r = img.shape[0]
    c = img.shape[1]
    dr = [[0, -1, -1], [-1, -1, 0], [0, 1, 1], [1, 1, 0]]
    dc = [[1, 1, 0], [0, -1, -1], [-1, -1, 0], [0, 1, 1]]
    with open('Yokoi.txt', 'w') as f:
        for i in range(r):
            for j in range(c):
                if (img[i][j] == 255):
                    R, q = 0, 0
                    for k in range(4):
                        cnt = 0
                        for l in range(3):
                            pr, pc = i+dr[k][1], j+dc[k][1]
                            if l == 0:
                                if ((pr < 0) or (pr >= r) or (pc < 0) or (pc >= c) or (img[pr][pc] != 255)):
                                    break
                                else:
                                    cnt = 1
                            else:
                                if ((pr >= 0 and pr < r and pc >= 0 and pc < c) and (img[pr][pc] == 255)):
                                    cnt += 1
                        if cnt == 3:
                            R += 1
                        elif 0 < cnt and cnt < 3:
                            q += 1
                    if R == 4:
                        f.write('5')
                    else:
                        f.write(str(q))
                else:
                    f.write(' ')
            f.write('\n')
```

```
def downSample(img):
    new_img = np.zeros((64,64), dtype=int)
    r = img.shape[0]
    c = img.shape[1]
    for i in range(0, r, 8):
        for j in range(0, c, 8):
            new_img[i//8][j//8] = img[i][j]
    return new_img
```

Result:

```
11111111 121111111111122322221 1111111111111 0 0
15555551 115555555511 2 11 11 115555555511 0
15555551 1 2115555112 21112221 15555555551 21
15555551 1 2 155112 22221511 155555555511 1
15555551 22 2112 22 121 0 0 155555555511 0
15555551 1 2 21 2 1 1 155555555511 0
15555551 12 1 121111 1321 155555555511
15111551 1322 115551111 155555555511
111 1551 1 12155555511 155555555511
11 1551 2115555511 155111555511
21 1551 2 15555555111 1551 1155511
1 1551 2 15555555511 1551 115551
1551 1121155555551 1551 15511
1551 15555555555511 1551 1111
1551 2221155555555511 1151 11
1551 22 1 15555555555511 151 11111
1551 2 1 115555555555551 151 115551
1551 2 11555555555555511511155511 115551
1551 12 115555555555555555551 155551
1551 11 0 2215555555555555555555112 1155551
1551 111 22 15555555555555555555551 1
1551 1511 1 1251121111121115555555111 1155551
1551 15521 1 121 1 11 1 15555555111 0 1555551
1551 1151 132 2 115555111 0 1155551
1551 151 0 322 115555111 121 1555551
1551 1221 2 1555551 131 1155551
1551 2 0 1 11555511 1 1155551
1551 2 0 0 115555551 0 1 1555551
1551 2 0 115555551 211555551
1551 1 0 1155555551 155555551
1551 1 1 11511115555521 1 115555551
1551 1111 1155511 2 1555555551
1551 131 111 15111 2 1555555551
1551 121 0 1121 1 111 1 2 1155555551
1551 11 111 1 221 11 1 2 1555555551
1551 12 0 1 21 121 11 1111 2 1555555551
1551 1 1 12 22 1511111111551 2 11555555551
1551 1 2 0 0 22 1555551115511 1 15555555551
1551 1 0 0 21 155551 1 151 2 155555555551
1551 1 0 0 2 15555112 151 2 155555555551
1551 1 1 1 115555511111 2 155555555551
1551 1 2 22 111511111212 2115555555551
1551 0 1 12 151 2 1 1555555511155551
1551 0 0 0 1111 121 155555551 1555551
1551 0 0 11111111 155555551 1555551
1551 15551 155555551 1555511
1551 211111111 155511
1 11521 1 12 122155511 2 11 115511
1 151 0 1 15555111 2111 15511
22 1511 1 15555555111 155111 1511
2 151 1 1555555551 155551 1151
2 1521 0 1115555555511 155511 1511
2 151 1555555555551 15551 12151
2 151 1555555555551 155511 1551
21 1511 0 1555555555551 1111111151
11 151 115555555555551 111511
11 151 155555555555551 151
11 151 115555555555551 211
11 151 115555555555551 1
11 151 0 155555555555551
11 111 1211111111111111111
```