

Computer Vision Homework 6

How to run it

```
python image.py lena.bmp
```

Principal code fragment

```
def hFunc( b, c, d, e ):
    if b == c and (d!=b or e!=b):
        return 'q'
    elif b == c and (d==b or e==b):
        return 'r'
    elif b != c:
        return 's'

def fFunc(a1,a2,a3,a4):
    neighbors = [a1,a2,a3,a4]
    if( neighbors.count('r') == len(neighbors) ): # all neighbors are equal to r
        return 5
    else:
        return neighbors.count('q')

def Yokoi( neighbors ):
    return fFunc( \
        hFunc(neighbors[0],neighbors[1],neighbors[6],neighbors[2]), \
        hFunc(neighbors[0],neighbors[2],neighbors[7],neighbors[3]), \
        hFunc(neighbors[0],neighbors[3],neighbors[8],neighbors[4]), \
        hFunc(neighbors[0],neighbors[4],neighbors[5],neighbors[1]), \
    )
```

Description

First, using 8x8 blocks and take topmost-left pixel to down sample origin image as 64x64 size image. (The sample image is left one)

Next, Search every pixel's neighbors to get 4-connected value, then if all neighbors are 'r', Yokoi' number is 5. Otherwise the number is equal to amount of 'q' in neighbors.



Result

```

111111111 121111111111122322221 11111111111111 0 0
155555551 115555555511 2 11 11 1155555555511 0
155555551 1 2115555112 21112221 15555555551 21
155555551 1 2 155112 22221511 155555555511 1
155555551 22 2112 22 121 0 0 1555555555511 0
155555551 1 2 21 2 1 1 1555555555551 0
155555551 12 1 121111 1321 15555555555511
15111551 1322 1155551111 1555555555551
111 1551 1 12155555511 15555555555511
11 1551 2115555511 1551115555511
21 1551 2 15555555111 1551 11555511
1 1551 2 155555555511 1551 115551 1
1551 112115555555551 1551 15511 12
1551 15555555555511 1551 1111 111
1551 1 222115555555555511 1151 11 1151
1551 2 22 1 1555555555555511 151 11111 1551
1551 2 1 11555555555555551 151 115551 11551
1551 2 115555555555555511511155511 115551
1551 12 11555555555555555555555555551 155551
1551 11 0 2215555555555555555555555555112 1155551
1551 111 22 15555555555555555555555555551 1 1555551
1551 1511 1 125112111112111555555555111 11555551
1551 15521 1 121 1 11 1 15555555111 0 15555551
1551 1151 132 2 1155555111 0 11555551
1551 151 0 322 115555111 121 15555551
1551 1221 2 1555551 131 115555551
1551 2 0 1 115555511 1 115555551
1551 2 0 0 1155555551 0 1 155555551
1551 2 1155555551 2115555551
1551 1 0 11555555551 1555555551
1551 1 11511115555521 1 11555555551
1551 1 1 11111 1155511 2 15555555551
1551 131 111 15111 2 15555555551
1551 121 0 1121 1 111 1 2 11555555551
1551 11 111 1 221 11 1 2 15555555551
1551 12 0 1 21 121 11 1111 2 15555555551
1551 1 12 22 151111111551 2 115555555551
1551 1 0 0 22 155555115511 1 1555555555551
1551 1 0 0 1 15555511 11511 2 1155555555551
1551 0 0 21 155551 1 151 2 1555555555551
1551 1 2 15555112 151 2 1555555555551
1551 1 1 1 115555511111 2 1555555555551
1551 2 22 11151111212 2115555555555551
1551 0 1 12 151 2 1 1555555511155551
1551 0 0 0 1111 121 155555551 1555551
1551 0 11111111 155555551 1555551
1551 0 115551 155555551 1555511
1551 15551 211111111 155511
11521 1 12 122155511 2 11 115511
1 151 0 1 155555111 2111 15511
22 1511 1 15555555111 155111 1511
22 1511 1 1555555551 155551 1151
2 151 0 1 11155555555511 15511 1511
2 1521 0 1 155555555555511 15551 12151
2 151 121 155555555555551 155511 1551
2 1511 0 155555555555551 115551 1511
21 1511 11 155555555555551 111111151
11 151 0 1155555555555551 111511
11 151 1555555555555551 151
11 151 0 11555555555555551 211
11 151 11555555555555551 1
11 151 0 15555555555555551
11 111 0 121111111111111111

```