

Computer Vision HW6: Yokoi Connectivity Number

R10741015 鄭傑鴻

Oct. 21, 2022

Original Image:



Step 1: Down Sampling

- Description:

Get the binary image by taking a threshold=128, and then take the topmost-left pixel in the 8x8 block.

- Algorithm:

binary_pic = original_pic >= 128 (Thresholding, using the broadcast property)

for each pixel:

*assign binary[i*8][j*8] to binary_downsampled[i][j] (Down sampling)*

- Code:

```
self.binary = (self.pic >= 128).astype(int)
self.binary_ds = np.zeros((64, 64)).astype(int)
self.downsample()

def downsample(self):
    for i in range(64):
        for j in range(64):
            self.binary_ds[i][j] = self.binary[i*8][j*8]
    self.printImg(self.binary_ds)
```

- Result:

Down sampled Binary Image (Value 0 replaced by '_' to better visualize):

```
11111111_____111111111111111111_____111111111111_____1_1
11111111_____111111111111_1_11_11_____111111111111_____1_
11111111_____1_1111111111_11111111_____111111111111_____11_
```


Return 'r' if $b==c$ and $(d==b \text{ and } e==b)$

Return 's' if $b!=c$

Function f:

Return 5 if a_1, a_2, a_3, a_4 are all 'r'

Else return the number of 'q's

For every pixel:

Calculate a_1, a_2, a_3, a_4 using the function h

Calculate the result using function f

Assign the result to $answer[i][j]$

- Code:

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

def funF(self, a1, a2, a3, a4):
    if a1=='r' and a2=='r' and a3=='r' and a4=='r':
        return '5'
    else:
        ret_val = 0
        if a1 == 'q': ret_val += 1
        if a2 == 'q': ret_val += 1
        if a3 == 'q': ret_val += 1
        if a4 == 'q': ret_val += 1
        return ret_val

def inRange(self, x, y):
    if x >= 64 or x<0:
        return False
    if y >= 64 or y<0:
        return False
    return True

def yokoi(self):
    for i in range(64):
        for j in range(64):
            if self.binary_ds[i][j] == 1:
                x0 = self.binary_ds[i][j]
                x1 = 0 if not self.inRange(i + 0, j + 1) else self.binary_ds[i + 0][j + 1]
                x2 = 0 if not self.inRange(i - 1, j + 0) else self.binary_ds[i - 1][j + 0]
                x3 = 0 if not self.inRange(i + 0, j - 1) else self.binary_ds[i + 0][j - 1]
                x4 = 0 if not self.inRange(i + 1, j + 0) else self.binary_ds[i + 1][j + 0]
                x5 = 0 if not self.inRange(i + 1, j + 1) else self.binary_ds[i + 1][j + 1]
                x6 = 0 if not self.inRange(i - 1, j + 1) else self.binary_ds[i - 1][j + 1]
                x7 = 0 if not self.inRange(i - 1, j - 1) else self.binary_ds[i - 1][j - 1]
                x8 = 0 if not self.inRange(i + 1, j - 1) else self.binary_ds[i + 1][j - 1]

                # Upper right: [0,0], [0, 1], [-1, 1], [-1, 0]
                a1 = self.funH(x0, x1, x6, x2)
                # Upper left: [0,0], [-1, 0], [-1, -1], [0, -1]
                a2 = self.funH(x0, x2, x7, x3)
                # Bottom left: [0,0], [0, -1], [1, -1], [1, 0]
                a3 = self.funH(x0, x3, x8, x4)
```

```

# Bottom right: [0,0], [1, 0], [1, 1], [0, 1]
a4 = self.funH(x0, x4, x5, x1)
self.answer[i][j] = self.funF(a1, a2, a3, a4)
else: self.answer[i][j] = 0
self.printImg(self.answer)

```

- Result:

```

11111111 12111111111122322221 111111111111
15555551 11555555511 2 11 11 11555555511
15555551 1 2115555112 21112221 15555555551 21
15555551 1 2 155112 22221511 155555555511 1
15555551 22 2112 22 121 155555555511
15555551 1 2 21 2 1 1 155555555551
15555551 12 1 121111 1321 15555555555511
15111551 1322 1155551111 1555555555551
111 1551 1 121555555511 15555555555511
11 1551 21155555511 1551115555511
21 1551 2 15555555111 1551 11555511
1 1551 2 155555555511 1551 115551 1
1551 112115555555551 1551 15511 12
1551 1555555555555511 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 115555555555555555111511155511 115551
1551 12 11555555555555555555555555551 155551
1551 11 2215555555555555555555555555112 1155551
1551 111 22 1555555555555555555555551 1 1555551
1551 1511 1 125112111111211155555555111 1155551
1551 15521 1 121 1 11 1 15555555111 1555551
1551 1151 132 2 11555555111 115555551
1551 151 322 115555111 121 155555551
1551 1221 2 1555551 131 115555551
1551 2 1 11555511 1 115555551
1551 2 115555551 1 155555551
1551 2 1155555551 21155555551
1551 1 11555555551 15555555551
1551 1 11511115555521 1 11555555551
1551 1 1 11111 1155511 2 15555555551
1551 131 111 15111 2 15555555551
1551 121 1121 1 111 1 2 115555555551
1551 11 111 1 221 11 1 2 155555555551
1551 12 1 21 121 11 1111 2 155555555551
1551 1 12 22 151111111551 2 1155555555551
1551 1 2 1555551115511 1 1555555555551
1551 2 22 12555551 15551 1 1555555555551
1551 1 1 1555511 11511 2 1155555555551
1551 21 155551 1 151 2 15555555555551
1551 2 15555112 151 2 15555555555551
1551 1 1 115555511111 2 15555555555551
1551 2 22 111511111212 211555555555551
1551 1 12 151 2 1 1555555511155551
1551 1111 121 15555551 1555551
1551 1111111 15555551 1555551
1551 115551 15555551 15555511
1551 15551 211111111 155511
11521 1 12 122155511 2 11 115511
1 151 1 1 155555111 2111 15511
22 1511 1 15555555111 155111 1511
22 1511 1 1555555551 155551 1151
2 151 1 11155555555511 155511 1511
2 1521 1 155555555555511 15551 12151
2 151 121 15555555555551 155511 1551
2 1511 155555555555551 115551 1511
21 1511 11 155555555555551 111111151
11 151 11555555555555551 111511
11 151 15555555555555551 151
11 151 11555555555555551 211
11 151 11555555555555551 1
11 151 15555555555555551
11 111 12111111111111111111

```