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:

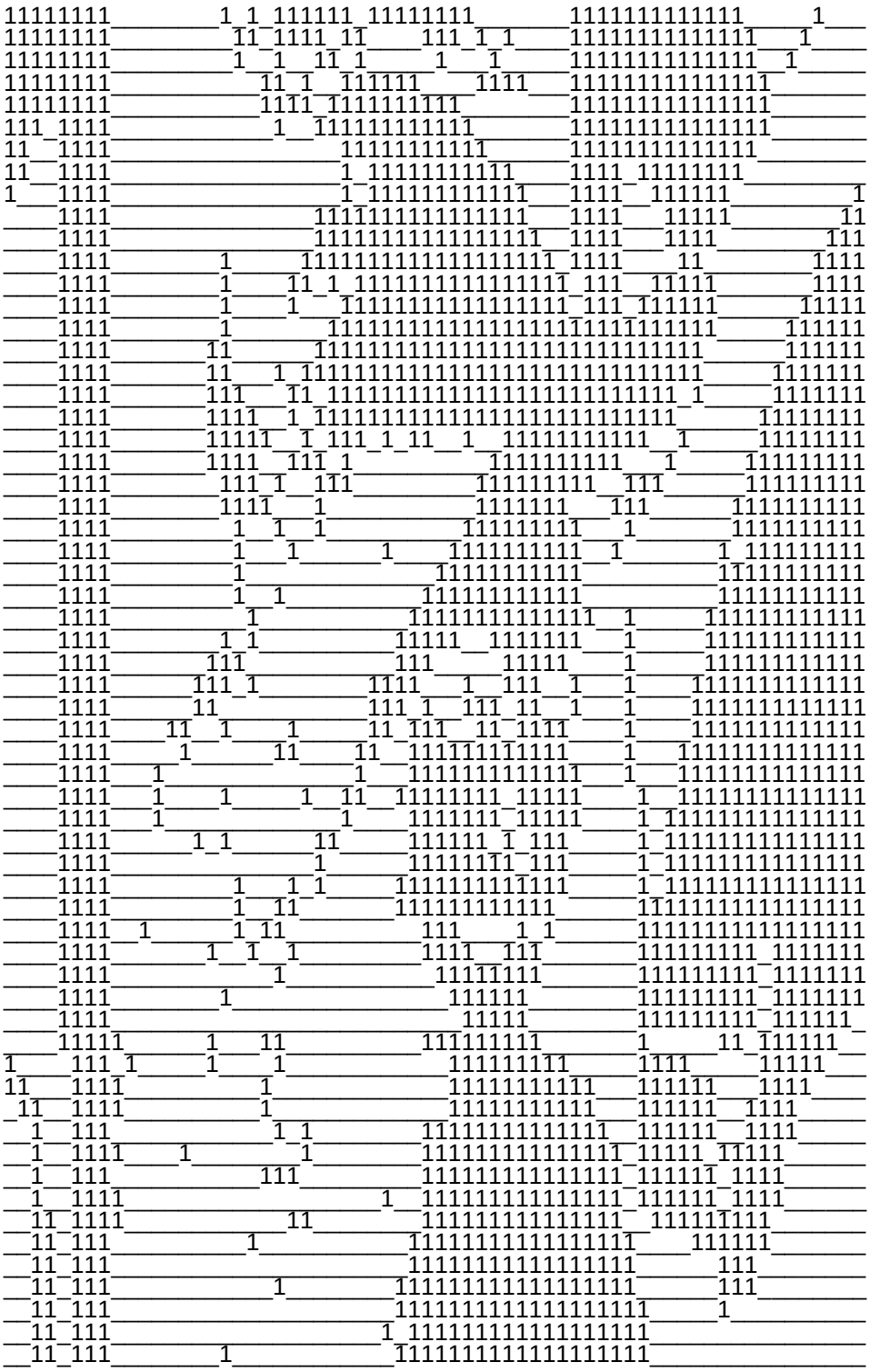
```
self.binary = (self.pic >= 128).astype(int)
self.binary_ds = np.zeros((64, 64)).astype(int)
self.downsample()

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________1111111111111111____111111111111______1__1
11111111________11111111111_1_11__11___111111111111________1_
11111111________1_1111111111__11111111____111111111111______11__
```



Step 2: Counts the Yokoi Connectivity Number:

- Description:

  Using the formulas on the slide/textbook, set up function *h* to get the intermediate results, and use function *f* to get the final result. Apply both functions to all pixels.

- Algorithm:

  Function h:

  Return ‘q’ if b==c and (d!=b or e!=b)

Return ‘r’ if b==c and (d==b and e==b)

Return ‘s’ if b!=c

Function f:

Return 5 if a1,a2,a3,a4 are all ‘r’

Else return the number of ‘q’s

For every pixel:

Calculate a1, a2, a3, a4 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: