Computer Vision

Home Work 6

Name – ROHIT DAS Student ID – 61047086s

Project –Yokoi connectivity Number.

Language and library used: Python, OpenCV, Pillow Numpy.

Description: This program will perform the following functions while executing lena.bmp image file:

1. Binarize Image
2. Down sample Image using 8*8 blocks
3. Find the Yokoi Connectivity number
4. Save it in text file

Parameters: None. Please Copy-paste the image path inside the program.

Algorithms Used –

Part 1: Binarize Image

Principal Code:

```
def img_binarize(img_in):

    image_binary = np.copy(img_in)

    for h_row in range(0, row):

        for w_col in range(0, col):

            if img_in[h_row,w_col] > 127:

                image_binary[h_row, w_col] = 255

            else:

                image_binary[h_row, w_col] = 0

    return (image_binary)
```

Part 2: Downsample using 8*8 blocks

Principal Code:

```
def downsampling(binary_img):

    img_down = np.zeros((64, 64), np.int)

    for h_row in range(0,img_down.shape[0]):

        for w_col in range(1,img_down.shape[1]):

            img_down[h_row][w_col] = binary_img[8 * h_row][8 * w_col]
    return img_down
```

Part 3: Yokoi Connectivity Number (4 neighbours)

Principal Code –

```
# using 4 connectivity algorithm
def four_connectivity_hFunction(b,c,d,e):
    # Find the connection pattern
    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 four_connectivity_fFunction(a1,a2,a3,a4):
    # Label the relation accordingly
    if ([a1, a2, a3, a4].count('r') == 4):
        # Return label 5 (interior).
        return 5
    else:
        # Return count of 'q'.
        # 0: Isolated, 1: Edge, 2: Connecting, 3: Branching, 4: Crossing.
        return [a1, a2, a3, a4].count('q')
# compute Yokoi Connectivity Number ...
def yokoi_connectivity_number(downsamples_image):
# Create a blank canvas of size 64*64 for sketching the Yokoi Number
    YokoiConnectivityNumber = np.full(downsamples_image.size, ' ')
    # Scan each column in original image.
    for h_row in range(downsamples_image.size[0]):
        # Scan each row in original image.
        for w_col in range(downsamples_image.size[1]):
            if (downsamples_image.getpixel((h_row,w_col)) != 0):

                # Get neighborhood pixel values.

                neighborhoodPixels = getNeighborhoodPixels(downsamples_image, h_row,w_col)

                # Calculating the pattern of relation between the neighboring pixel

                YokoiConnectivityNumber[h_row,w_col] = four_connectivity_fFunction(

                    four_connectivity_hFunction(neighborhoodPixels[0], neighborhoodPixels[1],

                                                neighborhoodPixels[6], neighborhoodPixels[2]),

                    four_connectivity_hFunction(neighborhoodPixels[0], neighborhoodPixels[2],

                                                neighborhoodPixels[7], neighborhoodPixels[3]),

                    four_connectivity_hFunction(neighborhoodPixels[0], neighborhoodPixels[3],

                                                neighborhoodPixels[8], neighborhoodPixels[4]),

                    four_connectivity_hFunction(neighborhoodPixels[0], neighborhoodPixels[4],

                                                neighborhoodPixels[5], neighborhoodPixels[1]))

            # This point is background.

            else:

                YokoiConnectivityNumber[h_row,w_col] = ' '



    # Return Yokoi Connectivity Number.

    return YokoiConnectivityNumber

def getNeighborhoodPixels(downImage, h_row,w_col):

    """

    Corners neighborhood (for corresponding ith values in x)

    x7,x2,x6

    x3,x0,x1

    x8,x4,x5

    """

    return [getValue(downImage, h_row, w_col), getValue(downImage, h_row + 1, w_col),
getValue(downImage, h_row, w_col - 1),

            getValue(downImage, h_row - 1, w_col), getValue(downImage, h_row, w_col + 1),
getValue(downImage, h_row + 1, w_col + 1),
            getValue(downImage, h_row + 1, w_col - 1), getValue(downImage, h_row - 1, w_col
- 1), getValue(downImage, h_row - 1, w_col + 1)]


def getValue(img, row, col):
    # Helper function to get the value separately for better code readability
    if row >= img.size[0] or row < 0 or col >= img.size[1] or col < 0:
        return 0
    return img.getpixel((row,col))
```

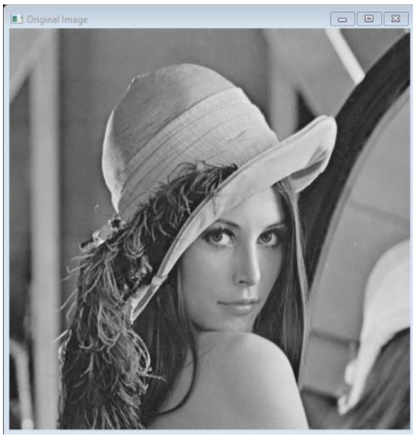
Part 4: Save it in .txt file

Principal Code-

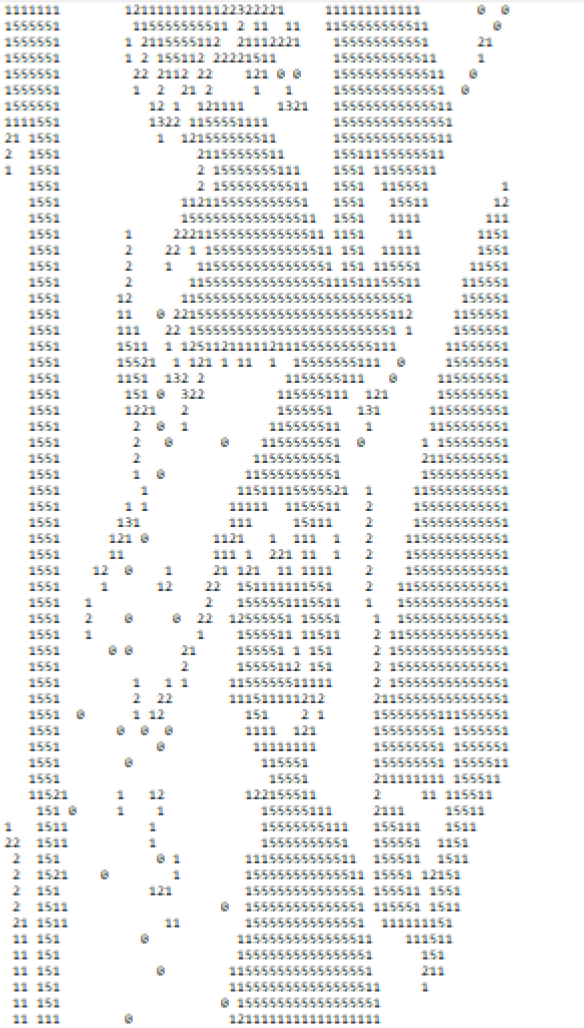
```
if __name__ == "__main__":
    # Converted the image to Binary in OpenCV for better readability of Code
    bin_image = img_binarize(gray_image)
    downsampled_cv_img = downsampling(bin_image)
    # Converted the OpenCV image to Pillow image for less datatype conversion constraints
and better handling og RG
    downsampled_pil_img = Image.fromarray(downsampled_cv_img)
    # Get Yokoi Connectivity Number.
    YokoiConnectivityNumber = yokoi_connectivity_number(downsampled_pil_img)
    # Save Yokoi Connectivity Number to file.
    np.savetxt('YokoiConnectivityNumber.txt',YokoiConnectivityNumber.T,delimiter='',
fmt='%s')
```

Example:

- Original image



- Yokoi Connectivity Number



```
1111111        12111111111122322221      111111111111       0  0
1555551         115555555511 2 11   11   1155555555511        0
1555551        1 2115555112  21112221     155555555551      21
1555551        1 2 155112 22221511        1555555555511     1
1555551         22 2112 22    121 0 0     15555555555511   0
1555551         1  2  21 2     1    1     15555555555551  0
1555551           12 1  121111     1321   155555555555511
1111551           1322 1155551111         155555555555551
21 1551            1  1215555555511       155555555555511
2  1551                 21155555511       15511155555511
1  1551                 2 15555555111     1551 11555511
   1551                 2 155555555511    1551  115551         1
   1551               1121155555555551    1551   15511        12
   1551               15555555555555511   1551   1111        111
   1551        1     2221155555555555511  1151    11        1151
   1551        2    22 1 1555555555555511  151  11111       1551
   1551        2    1   11555555555555551  151 115551      11551
   1551        2       115555555555555551115111155511     115551
   1551       12      115555555555555555555555555551      155551
   1551       11   0 2215555555555555555555555555112     1155551
   1551       111   22 155555555555555555555555551 1     1555551
   1551       1511  1 1251121111121115555555555111      11555551
   1551       15521  1 121 1 11  1   15555555111  0     15555551
   1551       1151  132 2           1155555111   0     115555551
   1551        151 0  322         115555111   121      155555551
   1551        1221   2           1555551    131      1155555551
   1551         2  0  1          115555511    1       1155555551
   1551         2   0      0    1155555551  0        1 155555551
   1551         2              11555555551           21155555551
   1551         1  0          115555555551           15555555551
   1551          1           11511115555521   1     115555555551
   1551        1 1          11111   1155511   2     155555555551
   1551       131           111      15111    2     155555555551
   1551      121 0        1121   1   111  1   2    1155555555551
   1551      11           111 1  221 11   1   2    1555555555551
   1551    12  0    1     21 121  11 1111     2    1555555555551
   1551     1      12    22  151111111551     2   11555555555551
   1551   1              2   1555551115511    1   15555555555551
   1551   2    0     0  22  12555551 15551     1  15555555555551
   1551   1             1    1555511 11511     2 115555555555551
   1551      0 0      21     155551 1 151      2 155555555555551
   1551               2      15555112 151      2 155555555555551
   1551         1   1 1     1155555511111      2 155555555555551
   1551         2  22       111511111212       21155555555555551
   1551  0      1 12          151     2 1      15555555111555551
   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          1221555511       2     11 115511
    151 0     1    1            155555111      2111     15511
1   1511          1             155555555111   155111   1511
22  1511          1             15555555551    155551  1151
 2  151            0 1        111555555555511  155511  1511
 2  1521    0        1        1555555555555511 15551 12151
 2  151           121         1555555555555551 155511 1551
 2  1511                   0  1555555555555551 115551 1511
 21 1511            11        1555555555555551  11111151
 11 151          0           115555555555555511    111511
 11 151                      155555555555555551      151
 11 151            0        1155555555555555551      211
 11 151                     11555555555555555511     1
 11 151                    0 1555555555555555551
 11 111        0            1211111111111111111
```