Homework 6 R08921005 黃國郡

Answer

Yokoi connectivity number

Homework 6 R08921005 黃國郡

Description

1. Problem Formation

   I. Binarize the benchmark image Lena as in HW2 by 128

   II. using 8x8 blocks as a unit, take the topmost-left pixel as the down-sampled data, Down-sample Lena from 512x512 to 64x64

   III. Count the Yokoi connectivity number

   IV. Result of this assignment is a 64x64 matrix. Please **align** the matrix within 1 single A4 page (using 4-connected).

2. Method of Algorithms

   For 4-connectivity:

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

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

3. 本次作業使用 python, IDE 為 Spyder

Homework 6　　R08921005 黃國郡

4. Source Code [HW6.py]說明如下

主程式:

1. 先import會用到的library
2. 將lena圖讀進來
3. 進行binary 處理
4. 進行downsample, 8個取一個值
5. 進行yokoi 演算法
6. 最後將結果存為txt檔

```
86  if __name__ == '__main__':
87      from PIL import Image
88      import numpy as np
89
90      lena = Image.open("lena.bmp")
91      binary_lena = lena.point(lambda x: 0 if x < 128 else 255, '1')
92      downsamplingImage = downsampling(binary_lena, 8)
93      downsamplingImage.save('downsampling.bmp')
94      YokoiConnectivityNumber = Yokoi(downsamplingImage)
95      np.savetxt('YokoiConnectivityNumber.txt',YokoiConnectivityNumber,delimiter='',fmt='%s')
```

1. 先import會用到的library
2. 取出sample後的大小
3. 建立新的output影像大小格式
4. 每sampleFactor的距離，將pixel取出來，放到output

```
1   # -*- coding: utf-8 -*-
2   """
3   Created on Sun Oct 20 10:32:58 2019
4
5   @author: vincent黃國郡
6   """
7   def downsampling(originalImage, sampleFactor):
8       """
9       originalImage: Image,sampleFactor: int, return Image
10      """
11      from PIL import Image
12      downsamplingWidth = int(originalImage.size[0] / sampleFactor)
13      downsamplingHeight = int(originalImage.size[1] / sampleFactor)
14      downsamplingImage = Image.new('1', (downsamplingWidth, downsamplingHeight))
15      for x in range(downsamplingWidth):
16          for y in range(downsamplingHeight):
17              originalPixel = originalImage.getpixel((x * sampleFactor, y * sampleFactor))
18              downsamplingImage.putpixel((x, y), originalPixel)
19      return downsamplingImage
20
21  def getNeighborhood(originalImage, position):
22      """
23      originalImage: Image,position: tuple, return  numpy array
24      """
25      neighborhood = np.zeros(9)
26      x, y = position
27      neighborhood = []
28      for xx in range(3):
29          for yy in range(3):
30              targetX = x + (xx - 1)
31              targetY = y + (yy - 1)
32              if ((0 <= targetX < originalImage.size[0]) and \
33                  (0 <= targetY < originalImage.size[1])):
34                  pixelValue = originalImage.getpixel((targetX, targetY))
35                  neighborhood.append(pixelValue)
36              else:
37                  neighborhood.append(0)
38      neighborhood = [
39          neighborhood[4], neighborhood[7], neighborhood[3],
40          neighborhood[1], neighborhood[5], neighborhood[8],
41          neighborhood[6], neighborhood[0], neighborhood[2]]
42      return neighborhood
```

1. 將每一個pixel 值附近的九宮個取出
2. 重新排列成yokoi會用到的九宮格順序

```python
def Yokoi_h(b, c, d, e):
    """
    type of  b,c,d,e: int  return: str
    """
    if ((b == c) and (b != d or b != e)):
        return 'q'
    if ((b == c) and (b == d and b == e)):
        return 'r'
    if (b != c):
        return 's'


def Yokoi_f(a1, a2, a3, a4):
    """
    type of  b,c,d,e: int  return: str
    0: Isolated 1: Edge 2: Connecting 3: Branching 4: Crossing 5:interior
    """
    if (a1 == a2 == a3 == a4 == 'r'):
        return 5
    else:
        return [a1, a2, a3, a4].count('q')

def Yokoi(originalImage):
    """
    type originalImage: Image ,return: numpy array
    """
    YokoiConnectivityNumber = np.full(downsamplingImage.size, ' ')

    for x in range(originalImage.size[0]):
        for y in range(originalImage.size[1]):
            if (originalImage.getpixel((x, y))):
                neighborhood = getNeighborhood(originalImage, (x, y))
                YokoiConnectivityNumber[y, x] = Yokoi_f(
                    Yokoi_h(neighborhood[0], neighborhood[1], neighborhood[6], neighborhood[2]),
                    Yokoi_h(neighborhood[0], neighborhood[2], neighborhood[7], neighborhood[3]),
                    Yokoi_h(neighborhood[0], neighborhood[3], neighborhood[8], neighborhood[4]),
                    Yokoi_h(neighborhood[0], neighborhood[4], neighborhood[5], neighborhood[1]))
            else:
                YokoiConnectivityNumber[y, x] = ' '

    return YokoiConnectivityNumber
```

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

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

進行 yokoi 運算，對每一個 pixel 進行

f(h1(a,b,c,d), h2(a,b,c,d).....)

最後回傳 array