NTNU 影像處理 HW2

廖家緯

2020.3.25

• Outline:

- (A) Given a grayscale image I,
- **Step 1:** Use the dithering matrix D_2 to generate an array D of image size by repeating

$$D_2$$
, where $D_2 = \begin{bmatrix} 0 & 128 & 32 & 160 \\ 192 & 64 & 224 & 96 \\ 48 & 176 & 16 & 144 \\ 240 & 112 & 208 & 80 \end{bmatrix}$.

- Step 2: Threshold image I by $I'(i,j) = \begin{cases} 255, & \text{if } I(i,j) > D(i,j) \\ 0, & \text{if } I(i,j) \leq D(i,j) \end{cases}$.
- **Step 3:** Show images I and I'.
- (B) Extend to n = 4 gray values

1.
$$\frac{255}{3} = 85$$
.

2.
$$Q(i,j) = \left\lceil \frac{I(i,j)}{85} \right\rceil$$
.

$$3. \ D_1 = \left[\begin{array}{cc} 0 & 56 \\ 84 & 28 \end{array} \right] \Longrightarrow D.$$

4.
$$I'(i,j) = Q(i,j) + \begin{cases} 1, & \text{if } I(i,j) - 85Q(i,j) > D(i,j) \\ 0, & \text{if } I(i,j) - 85Q(i,j) \le D(i,j) \end{cases}$$

5. Scale values of I' so that its values are in [0, 255] for displaying.

• Code(Python):

```
# coding: utf-8
   #引入模組
   import numpy as np
   import cv2
   #(A)讀取灰階影像
   I = cv2.imread('Gray.jpg', cv2.IMREAD_GRAYSCALE)
   n = np.shape(I)[0]
10
   m = np.shape(I)[1]
11
   #Step1:建構dithering matrix
   D2 = np.array([[0, 128, 32, 160], [192, 64, 224, 96], [48, 176, 16,
       144], [240, 122, 208, 80]])
15
   #用D2貼滿D
16
   D = np.zeros((n,m), np.int)
17
   for i in range(n):
       for j in range(m):
          D[i, j] = D2[(i\%4), (j\%4)]
20
   I1 = np.zeros((n,m), np.int)
21
22
   #Step2:
   for i in range(n):
       for j in range(m):
          if I[i, j] > D[i, j]:
26
             I1[i, j] = 255
2.7
          else:
28
              I1[i,j] = 0
   I1 = np.uint8(I1)
32
33
   #Step3:
   # 顯示原圖I
34
   cv2.imshow('My Image1', I)
   # 按下任意鍵則關閉所有視窗
   cv2.waitKey(0)
   cv2.destroyAllWindows()
39
40
   # 顯示I1
   cv2.imshow('My Image2', I1)
   # 按下任意鍵則關閉所有視窗
44
   cv2.waitKey(0)
   cv2.destroyAllWindows()
   Q = np.zeros((n,m), np.int)
```

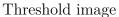
```
\#(B)Extend to n = 4 gray values
50
51
    #1.
52
   N = np.int(255/3)
53
54
    #2.
   Q = np.zeros((n,m), np.int)
57
    for i in range(n):
58
       for j in range(m):
59
           Q[i, j] = I[i, j]/N
60
61
    #3.建構dithering matrix
   D1 = np.array([[0, 56], [84, 28]])
63
64
    #extend to D
65
   D = np.zeros((n,m), np.int)
    for i in range(n):
       for j in range(m):
           D[i, j] = D1[(i\%2), (j\%2)]
69
70
71
    I2 = np.zeros((n,m), np.int)
72
    for i in range(n):
74
       for j in range(m):
75
           if I[i, j] - N*Q[i, j] > D[i, j]:
76
               I2[i, j] = Q[i, j] + 1
77
           else:
78
               I2[i, j] = Q[i, j]
80
    #5.
81
    for i in range(n):
82
       for j in range(m):
83
           I2[i,j] *= N
84
    I2 = np.uint8(I2)
86
    # 顯示12
87
    cv2.imshow('My Image2', I2)
88
89
    # 按下任意鍵則關閉所有視窗
    cv2.waitKey(0)
    cv2.destroyAllWindows()
92
93
94
95
   #儲存影像
    cv2.imwrite('I.jpg', I)
    cv2.imwrite('I1.jpg', I1)
    cv2.imwrite('I2.jpg', I2)
98
```

• Result:



原圖 (灰階影像)







n = 4

• Experience:

如何將矩陣 D_2 放大至 D,這裡我嘗試用不同方法,然後 Python 不像 Matlab 有許多好用的函式,最後我還是用 for 迴圈掃過每個元素,搭配 mod 運算使用。這樣可能提高程式執行的時間複雜度,增加 $O(n^2)$ 。而我在寫這支程式的時候是依照老師給的步驟一步步做出來,在我完成之後,了解到這次作業意義: 我們將一灰階相片改成用兩個值儲存或著 4 個值儲存的方法。