

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

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

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. $Q(i, j) = \left\lfloor \frac{I(i, j)}{85} \right\rfloor.$

3. $D_1 = \begin{bmatrix} 0 & 56 \\ 84 & 28 \end{bmatrix} \Rightarrow D.$

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

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

- Code(Python):

```
1  # coding: utf-8
2
3  #引入模組
4  import numpy as np
5  import cv2
6
7
8  #(A)讀取灰階影像
9  I = cv2.imread('Gray.jpg', cv2.IMREAD_GRAYSCALE)
10 n = np.shape(I)[0]
11 m = np.shape(I)[1]
12
13 #Step1:建構dithering matrix
14 D2 = np.array([[0, 128, 32, 160], [192, 64, 224, 96], [48, 176, 16,
15     144], [240, 122, 208, 80]])
16
17 #用D2貼滿D
18 D = np.zeros((n,m), np.int)
19 for i in range(n):
20     for j in range(m):
21         D[i, j] = D2[(i%4), (j%4)]
22
23 I1 = np.zeros((n,m), np.int)
24
25 #Step2:
26 for i in range(n):
27     for j in range(m):
28         if I[i, j] > D[i, j]:
29             I1[i, j] = 255
30         else:
31             I1[i, j] = 0
32
33 I1 = np.uint8(I1)
34
35 #Step3:
36 # 顯示原圖I
37 cv2.imshow('My Image1', I)
38
39 # 按下任意鍵則關閉所有視窗
40 cv2.waitKey(0)
41 cv2.destroyAllWindows()
42
43 # 顯示I1
44 cv2.imshow('My Image2', I1)
45
46 # 按下任意鍵則關閉所有視窗
47 cv2.waitKey(0)
48 cv2.destroyAllWindows()
49 Q = np.zeros((n,m), np.int)
```

```

49
50 # (B) Extend to n = 4 gray values
51
52 #1.
53 N = np.int(255/3)
54
55 #2.
56 Q = np.zeros((n,m), np.int)
57
58 for i in range(n):
59     for j in range(m):
60         Q[i, j] = I[i, j]/N
61
62 #3. 建構dithering matrix
63 D1 = np.array([[0, 56], [84, 28]])
64
65 #extend to D
66 D = np.zeros((n,m), np.int)
67 for i in range(n):
68     for j in range(m):
69         D[i, j] = D1[(i%2), (j%2)]
70
71 #4.
72 I2 = np.zeros((n,m), np.int)
73
74 for i in range(n):
75     for j in range(m):
76         if I[i, j] - N*Q[i, j] > D[i, j]:
77             I2[i, j] = Q[i, j] + 1
78         else:
79             I2[i, j] = Q[i, j]
80
81 #5.
82 for i in range(n):
83     for j in range(m):
84         I2[i,j] *= N
85 I2 = np.uint8(I2)
86
87 # 顯示I2
88 cv2.imshow('My Image2', I2)
89
90 # 按下任意鍵則關閉所有視窗
91 cv2.waitKey(0)
92 cv2.destroyAllWindows()
93
94
95 #儲存影像
96 cv2.imwrite('I.jpg', I)
97 cv2.imwrite('I1.jpg', I1)
98 cv2.imwrite('I2.jpg', I2)

```

- **Result:**



原圖 (灰階影像)



Threshold image



$n = 4$

- **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。