

数字图像处理作业5

姓名：吴宇祺 学号：16340242

第一题

算法描述：

1. 零扩展：将矩阵A左边扩展一列0，右边两列0，底部一行0

```
1 | expendA = zeros(M+1, N+3); % 零扩展A，下补一行，左补一列，右补一列
2 | expendA(1:M, 2:N+1) = A;
```

- 使用模板1进行膨胀得到矩阵B：与模板进行点乘的结果如果大于等于1，说明当前区域中存在与模板1对应的像素，满足膨胀条件，置为1。

```
1 | for i = 1:M
2 |     for j = 2:N+1
3 |         tmp = expendA(i, j:j+2); %截取当前模板对应区域
4 |         if sum(tmp.*mask1) >= 1
5 |             B(i, j-1) = 1;
6 |         end
7 |     end
8 | end
```

- 使用模板1进行腐蚀得到矩阵C：与模板进行点乘的结果如果不等于3，说明当前区域中存在与模板1不对应的像素（即当前区域不全为1），满足腐蚀条件，置为0，否则为。

```
1 | for i = 1:M
2 |     for j = 2:N+1
3 |         tmp = expendA(i, j:j+2);
4 |         if sum(tmp.*mask1) == 3
5 |             C(i, j-1) = 1;
6 |         end
7 |     end
8 | end
```

- 使用模板2进行膨胀得到矩阵D

```
1 | for i = 1:M
2 |     for j = 2:N+1
3 |         tmp = expendA(i:i+1, j-1:j);
4 |         if sum(sum(tmp.*mask2)) >= 1
5 |             D(i, j-1) = 1;
6 |         end
7 |     end
8 | end
```

- 使用模板2进行腐蚀得到矩阵E：与模板进行点乘的结果如果等于3，说明当前区域中存在大于等于3个1点，在此条件下左下像素为0，满足腐蚀的保留像素条件，置为1

```

1  for i = 1:M
2      for j = 2:N+1
3          tmp = expendA(i:i+1, j-1:j);
4          if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
5              E(i,j-1) = 1;
6          end
7      end
8  end

```

- 使用模板1进行开运算（先腐蚀后膨胀）得到矩阵F
- 使用模板2进行开运算得到矩阵G
- 使用模板1进行闭运算（先腐蚀后膨胀）得到矩阵E
- 使用模板2进行闭运算得到矩阵F

测试结果

- 输入给定矩阵A

A =

0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	1	0	0
0	0	1	1	1	1	0
0	0	1	1	1	0	0
0	1	0	1	0	1	0
0	0	0	0	0	0	0

- 使用模板1进行膨胀得到矩阵B

B =

0	0	0	0	0	0	0
1	1	1	1	0	0	0
0	1	1	1	0	0	0
0	1	1	1	1	0	0
1	1	1	1	1	1	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0
0	0	0	0	0	0	0

- 使用模板1进行腐蚀得到矩阵C

C =

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

- 使用模板2进行膨胀得到矩阵D

D =

0	0	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	0	0
0	0	1	1	1	1	0
0	0	1	1	1	1	1
0	1	1	1	1	1	0
0	1	1	1	1	1	1
0	0	0	0	0	0	0

- 使用模板2进行腐蚀得到矩阵E

E =

0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

- 使用模板1进行开运算（先腐蚀后膨胀）得到矩阵F

F =

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1	1	1	1	0	0	0
1	1	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

- 使用模板2进行开运算得到矩阵G

G =

0	0	0	1	0	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

- 使用模板1进行闭运算得到矩阵H

H =

0	0	0	0	0	0	0
1	1	0	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	1	1	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0
0	0	0	0	0	0	0

- 使用模板2进行闭运算得到矩阵I

I =

0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

第二题

算法描述

基本全局阈值实际上和k=2时的K-mean聚类算法一致，用了后者实现。

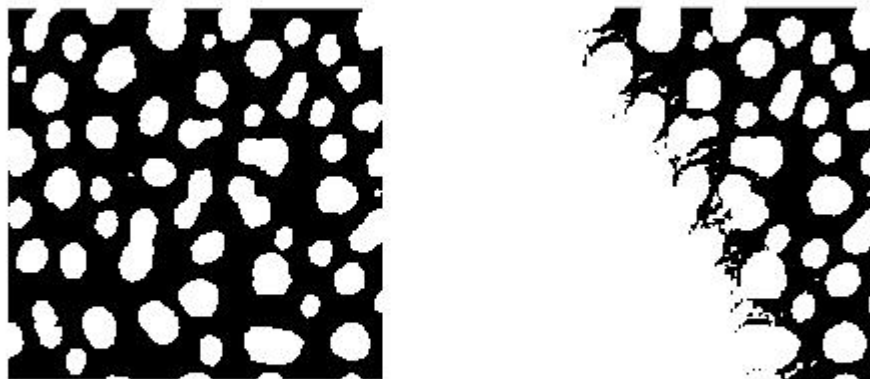
1. 初始化u1, u2
2. 误差大于给定值eps时，进行迭代：
 1. 遍历像素，计算当前像素灰度值到u1和u2的距离（绝对值）。若距离u1较近归类为1，否则归类为0。
 2. 计算新的u1, u2分别为步骤1中类1和类0的像素的灰度值的均值
3. 分类完成

核心代码

```
1 while loop <= maxLoop && (abs(u1 - last_u1) > eps || abs(u2 - last_u2) > eps)
2     last_u1 = u1;
3     last_u2 = u2;
4     sum1 = 0;
5     sum2 = 0;
6     Rnk = zeros(M,N);
7
8     for i = 1:M
9         for j = 1:N
10             if abs(u1 - data(i,j)) < abs(u2 - data(i,j))
11                 sum1 = sum1+data(i,j);
12                 Rnk(i,j)=1;
13             else
14                 sum2 = sum2+data(i,j);
15                 Rnk(i,j)=0;
16             end
17         end
18     end
19     count = sum(sum(Rnk));
20     u1 = sum1/count;
21     u2 = sum2/(M*N-count);
22
23     loop = loop +1;
24 end
```

测试结果

观察到当图像亮度变化均匀时，算法效果好。



源代码

第一题

```
1  A = [0,0,0,0,0,0,0;
2       0,0,1,1,0,0,0;
3       0,0,0,1,0,0,0;
4       0,0,0,1,1,0,0;
5       0,0,1,1,1,1,0;
6       0,0,1,1,1,0,0;
7       0,1,0,1,0,1,0;
8       0,0,0,0,0,0,0];
9  mask1 = [1, 1, 1];
10 mask2 = [1, 1; 0, 1];
11 [M, N] = size(A);
12 mask1
13 expendA = zeros(M+1, N+3); % 零扩展A，下补一行，左补一列，右补一列
14 expendA(1:M, 2:N+1) = A;
15 A
16
17 %% dilation with mask1
18 B = zeros(M, N);
19 for i = 1:M
```

```

20     for j = 2:N+1
21         tmp = expendA(i, j:j+2);
22         if sum(tmp.*mask1)>=1
23             B(i,j-1) = 1;
24         end
25     end
26 end
27 B
28
29 %% erosion with mask1
30 C = zeros(M, N);
31 for i = 1:M
32     for j = 2:N+1
33         tmp = expendA(i, j:j+2);
34         if sum(tmp.*mask1) == 3
35             C(i,j-1) = 1;
36         end
37     end
38 end
39 C
40
41 %% dilation with mask2
42
43 D = zeros(M, N);
44 for i = 1:M
45     for j = 2:N+1
46         tmp = expendA(i:i+1, j-1:j);
47         if sum(sum(tmp.*mask2)) >= 1
48             D(i,j-1) = 1;
49         end
50     end
51 end
52 D
53
54 %% erosion with mask2
55
56 E = zeros(M, N);
57 v = sum(sum(mask2));
58 for i = 1:M
59     for j = 2:N+1
60         tmp = expendA(i:i+1, j-1:j);
61         if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
62             E(i,j-1) = 1;
63         end
64     end
65 end
66 E
67
68 %% opening with mask1
69 F0 = zeros(M, N);
70 % erosion
71 for i = 1:M
72     for j = 2:N+1

```

```

73     tmp = expendA(i, j:j+2);
74     if sum(tmp.*mask1) == 3
75         F0(i,j-1) = 1;
76     end
77 end
78 end
79 % dilation
80 expendF0 = zeros(M+1, N+3); % 零扩展F0, 下补一行, 左补一列, 右补一列
81 expendF0(1:M, 2:N+1) = F0;
82 F = zeros(M, N);
83 for i = 1:M
84     for j = 2:N+1
85         tmp = expendF0(i, j:j+2);
86         if sum(tmp.*mask1) >= 1
87             F(i,j-1) = 1;
88         end
89     end
90 end
91 F
92
93 %% opening with mask2
94 G0 = zeros(M,N);
95 %erosion
96 for i = 1:M
97     for j = 2:N+1
98         tmp = expendA(i:i+1, j-1:j);
99         if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
100             G0(i,j-1) = 1;
101         end
102     end
103 end
104 %dilation
105 expendG0 = zeros(M+1, N+3); % 零扩展F0, 下补一行, 左补一列, 右补一列
106 expendG0(1:M, 2:N+1) = G0;
107 G = zeros(M,N);
108 for i = 1:M
109     for j = 2:N+1
110         tmp = expendG0(i:i+1, j-1:j);
111         if sum(sum(tmp.*mask2)) >= 1
112             G(i,j-1) = 1;
113         end
114     end
115 end
116 G
117
118
119 %% closing with mask1
120 %dilation
121 H0 = zeros(M, N);
122 for i = 1:M
123     for j = 2:N+1
124         tmp = expendA(i, j:j+2);
125         if sum(tmp.*mask1) >= 1

```



```

126         H0(i,j-1) = 1;
127     end
128 end
129 end
130 %erosion
131 expendH0 = zeros(M+1, N+3); % 零扩展F0，下补一行，左补一列，右补一列
132 expendH0(1:M, 2:N+1) = H0;
133 H = zeros(M, N);
134 for i = 1:M
135     for j = 2:N+1
136         tmp = expendH0(i, j:j+2);
137         if sum(tmp.*mask1) == 3
138             H(i,j-1) = 1;
139         end
140     end
141 end
142 H
143
144 %% closing with mask2
145 %dilation
146 I0 = zeros(M,N);
147 for i = 1:M
148     for j = 2:N+1
149         tmp = expendA(i:i+1, j-1:j);
150         if sum(sum(tmp.*mask2)) >= 1
151             I0(i,j-1) = 1;
152         end
153     end
154 end
155 expendI0 = zeros(M+1, N+3); % 零扩展H0，下补一行，左补一列，右补一列
156 expendI0(1:M, 2:N+1) = I0;
157
158 %erosion
159 I = zeros(M, N);
160 for i = 1:M
161     for j = 2:N+1
162         tmp = expendI0(i:i+1, j-1:j);
163         if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
164             I(i,j-1) = 1;
165         end
166     end
167 end
168 I
169

```

第二题

problem.m

```

1 close all;
2
3 img = imread(' ../blobz1.png');
4 img2 = imread(' ../blobz2.png');
5 %% kmean
6 kmean1 = Kmean2(img);
7 kmean2 = Kmean2(img2);
8 subplot(1,2,1), imshow(kmean1);
9 subplot(1,2,2), imshow(kmean2);

```

Kmean2.m

```

1 function[result] = Kmean2(img)
2 data = uint8(img);
3 [M,N] = size(img);
4
5 Rnk = zeros(M,N);
6 u1 = 0;
7 u2 = 125;
8 last_u1 = 1;
9 last_u2 = 126;
10
11 eps = 0.01;
12 maxLoop = 10;
13 loop = 0;
14
15 while loop <= maxLoop && (abs(u1 - last_u1) > eps || abs(u2 - last_u2) > eps)
16     last_u1 = u1;
17     last_u2 = u2;
18     sum1 = 0;
19     sum2 = 0;
20     Rnk = zeros(M,N);
21
22     for i = 1:M
23         for j = 1:N
24             if abs(u1 - data(i,j)) < abs(u2 - data(i,j))
25                 sum1 = sum1+data(i,j);
26                 Rnk(i,j)=1;
27             else
28                 sum2 = sum2+data(i,j);
29                 Rnk(i,j)=0;
30             end
31         end
32     end
33     count = sum(sum(Rnk));
34     u1 = sum1/count;
35     u2 = sum2/(M*N-count);
36
37     loop = loop +1;
38 end
39
40 result = 255*Rnk;

```

