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

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

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

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

- 使用模板2进行膨胀得到矩阵D

```
for i = 1:M
    for j = 2:N+1
       tmp = expendA(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) >= 1
           D(i,j-1) = 1;
       end
    end
end
```

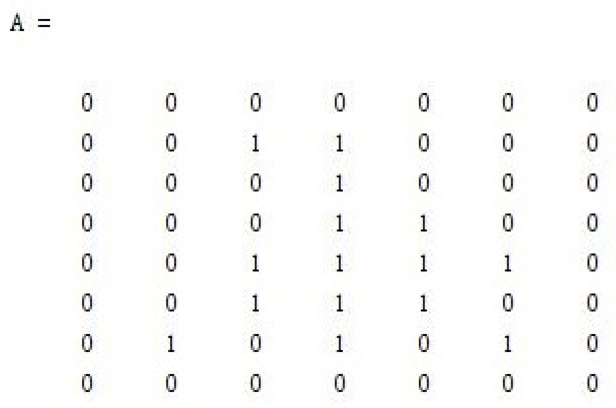
- 使用模板2进行腐蚀得到矩阵E：与模板进行点乘的结果如果等于3，说明当前区域中存在大于等于3个1点，在此条件下左下像素为0，满足腐蚀的保留像素条件，置为1

```
for i = 1:M
    for j = 2:N+1
        tmp = expendA(i:i+1, j-1:j);
        if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
            E(i,j-1) = 1;
        end
    end
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

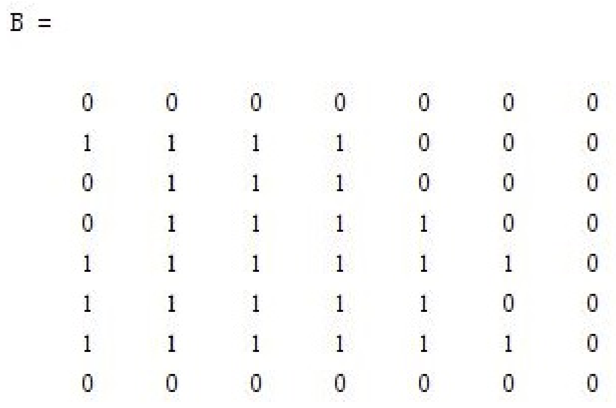
测试结果

- 输入给定矩阵A



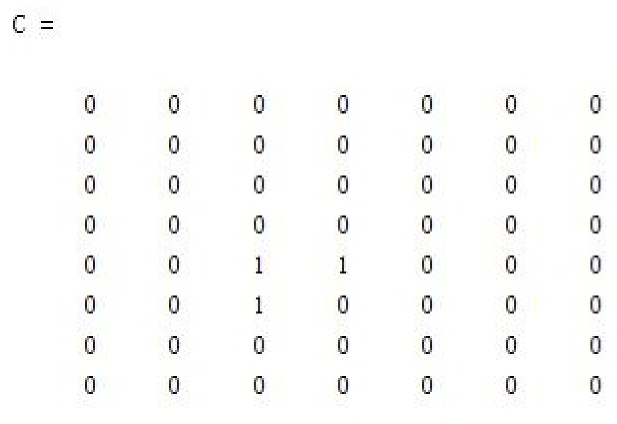
```
A =

     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



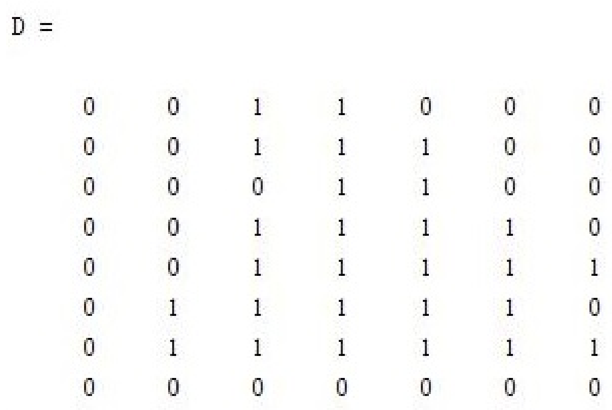
```
B =

     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



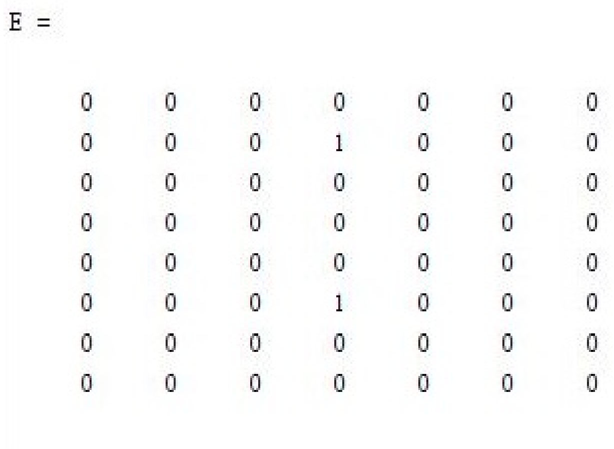
```
C =

     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



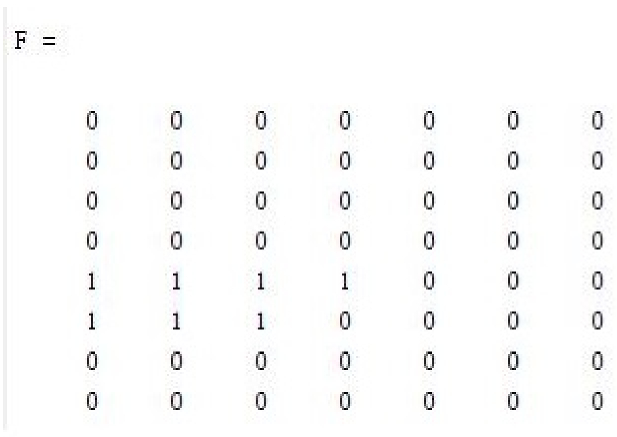
```
D =

     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



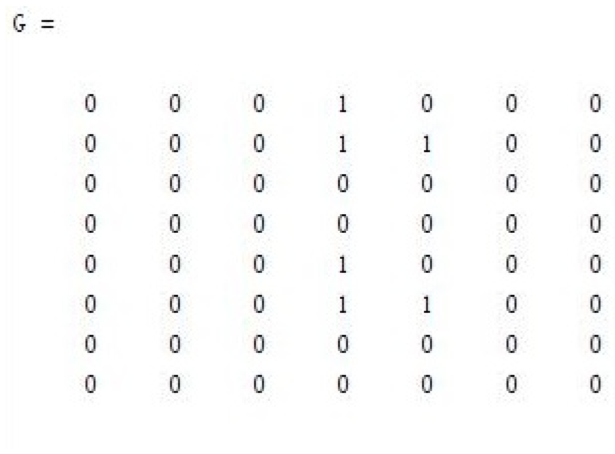
```
E =

     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



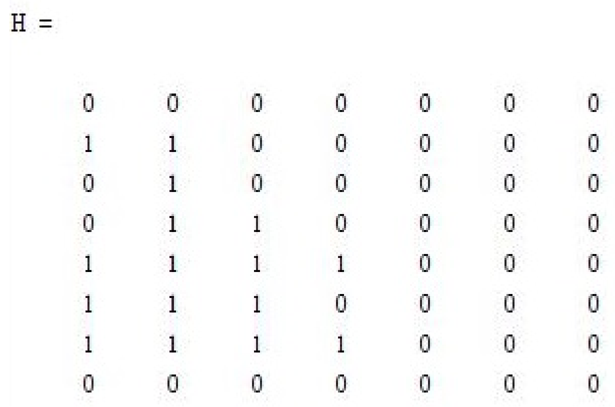
```
F =

     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



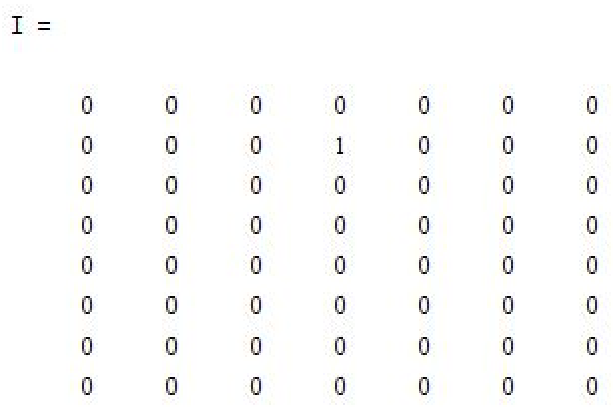
```
G =

     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。

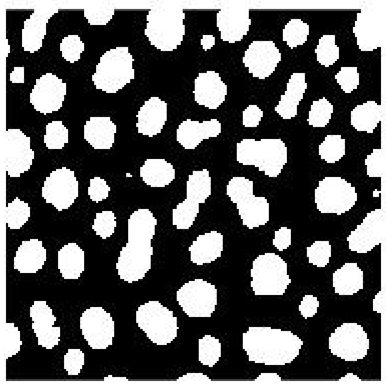
1. 初始化u1，u2
2. 误差大于给定值eps时，进行迭代：
   1. 遍历像素，计算当前像素灰度值到u1和u2的距离（绝对值）。若距离u1较近归类为1，否则归类为0。
   2. 计算新的u1，u2分别为步骤1中类1和类0的像素的灰度值的均值
3. 分类完成

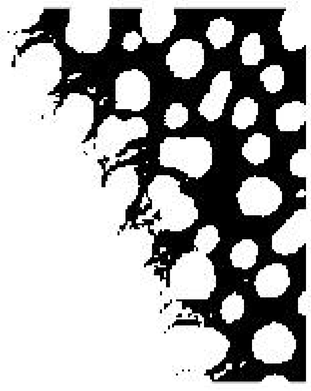
核心代码

```matlab
while loop <= maxLoop && (abs(u1 - last_u1) > eps || abs(u2 - last_u2) > eps)
    last_u1 = u1;
    last_u2 = u2;
    sum1 = 0;
    sum2 = 0;
    Rnk = zeros(M,N);

    for i = 1:M
        for j = 1:N
            if abs(u1 - data(i,j)) < abs(u2 - data(i,j))
                sum1 = sum1+data(i,j);
                Rnk(i,j)=1;
            else
                sum2 = sum2+data(i,j);
                Rnk(i,j)=0;
            end
        end
    end
    count = sum(sum(Rnk));
    u1 = sum1/count;
    u2 = sum2/(M*N-count);

    loop = loop +1;
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

```matlab
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];
mask1 = [1, 1, 1];
mask2 = [1, 1; 0, 1];
[M, N] = size(A);
mask1
expendA = zeros(M+1, N+3);  % 零扩展A，下补一行，左补一列，右补一列
expendA(1:M, 2:N+1) = A;
A

%% dilation with mask1
B = zeros(M, N);
 for i = 1:M
```

```
     for j = 2:N+1
        tmp = expendA(i, j:j+2);
        if sum(tmp.*mask1)>=1
            B(i,j-1) = 1;
        end
     end
 end
B

 %% erosion with mask1
 C = zeros(M, N);
 for i = 1:M
     for j = 2:N+1
        tmp = expendA(i, j:j+2);
        if sum(tmp.*mask1) == 3
            C(i,j-1) = 1;
        end
     end
 end
C

%% dilation with mask2

D = zeros(M, N);
for i = 1:M
    for j = 2:N+1
       tmp = expendA(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) >= 1
           D(i,j-1) = 1;
       end
    end
end
D

%% erosion with mask2

E = zeros(M, N);
v = sum(sum(mask2));
for i = 1:M
    for j = 2:N+1
       tmp = expendA(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
           E(i,j-1) = 1;
       end
    end
end
E

%% opening with mask1
FO = zeros(M, N);
% erosion
 for i = 1:M
     for j = 2:N+1
```

```matlab
            tmp = expendA(i, j:j+2);
            if sum(tmp.*mask1) == 3
                F0(i,j-1) = 1;
            end
        end
    end
% dilation
expendF0 = zeros(M+1, N+3);  % 零扩展F0，下补一行，左补一列，右补一列
expendF0(1:M, 2:N+1) = F0;
F = zeros(M, N);
 for i = 1:M
     for j = 2:N+1
        tmp = expendF0(i, j:j+2);
        if sum(tmp.*mask1) >= 1
            F(i,j-1) = 1;
        end
     end
 end
F

%% opening with mask2
G0 = zeros(M,N);
%erosion
for i = 1:M
    for j = 2:N+1
       tmp = expendA(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
           G0(i,j-1) = 1;
       end
    end
end
%dilation
expendG0 = zeros(M+1, N+3);  % 零扩展F0，下补一行，左补一列，右补一列
expendG0(1:M, 2:N+1) = G0;
G = zeros(M,N);
for i = 1:M
    for j = 2:N+1
       tmp = expendG0(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) >= 1
           G(i,j-1) = 1;
       end
    end
end
G


%% closing with mask1
%dilation
H0 = zeros(M, N);
 for i = 1:M
     for j = 2:N+1
        tmp = expendA(i, j:j+2);
        if sum(tmp.*mask1) >= 1
```

```
                H0(i,j-1) = 1;
            end
        end
    end
%erosion
expendH0 = zeros(M+1, N+3);  % 零扩展F0，下补一行，左补一列，右补一列
expendH0(1:M, 2:N+1) = H0;
H = zeros(M, N);
 for i = 1:M
     for j = 2:N+1
        tmp = expendH0(i, j:j+2);
        if sum(tmp.*mask1) == 3
            H(i,j-1) = 1;
        end
     end
 end
 H

%% closing with mask2
%dilation
I0 = zeros(M,N);
for i = 1:M
    for j = 2:N+1
       tmp = expendA(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) >= 1
           I0(i,j-1) = 1;
       end
    end
end
expendI0 = zeros(M+1, N+3);  % 零扩展H0，下补一行，左补一列，右补一列
expendI0(1:M, 2:N+1) = I0;

%erosion
I = zeros(M, N);
for i = 1:M
    for j = 2:N+1
       tmp = expendI0(i:i+1, j-1:j);
       if sum(sum(tmp.*mask2)) == v && tmp(2, 1) == 0
           I(i,j-1) = 1;
       end
    end
end
I

```

第二题

problem.m

```matlab
close all;

img = imread('../blobz1.png');
img2 = imread('../blobz2.png');
%% kmean
kmean1 = Kmean2(img);
kmean2 = Kmean2(img2);
subplot(1,2,1), imshow(kmean1);
subplot(1,2,2), imshow(kmean2);
```

Kmean2.m

```matlab
function[result] = Kmean2(img)
data = uint8(img);
[M,N] = size(img);

Rnk = zeros(M,N);
u1 = 0;
u2 = 125;
last_u1 = 1;
last_u2 = 126;

eps = 0.01;
maxLoop = 10;
loop = 0;

while loop <= maxLoop && (abs(u1 - last_u1) > eps || abs(u2 - last_u2) > eps)
    last_u1 = u1;
    last_u2 = u2;
    sum1 = 0;
    sum2 = 0;
    Rnk = zeros(M,N);

    for i = 1:M
        for j = 1:N
            if abs(u1 - data(i,j)) < abs(u2 - data(i,j))
                sum1 = sum1+data(i,j);
                Rnk(i,j)=1;
            else
                sum2 = sum2+data(i,j);
                Rnk(i,j)=0;
            end
        end
    end
    count = sum(sum(Rnk));
    u1 = sum1/count;
    u2 = sum2/(M*N-count);

    loop = loop +1;
end

result = 255*Rnk;
```