Report of HW5

夏浩 19307130268

[HW5-1] 实现K类均值分类的分割算法或基于高斯混合模型的分割算法（备注：二选一就行，不可以调用别的库实现的函数），并使用噪声污染过的图像（如P=0.1%的椒盐噪声）测试一下算法：

（1）测试二类分割，并对比自己实现的算法的分割结果与阈值算法（如OSTU或基于最大熵）二值化的结果；

（2）测试多类（大于等于三类）分割（请自己设定分割标签类别的个数）；

（3）针对噪声图像，讨论为什么分割的结果不准确，有什么方法可以取得更好的分割结果（备注：不要求实现该方法，只是讨论）。

代码实现：

Kmeans:

```
clear
clc

%读取原图和部分噪声图
ori = 'Ori\';
ori_img10 = double(imread([ori,'brain.png']));
[r,c] = size(ori_img10)
ori_img20 = double(imread([ori,'heart.png']));
ori_img13 = double(imread([ori,'brain+salt&pepper.png']));
ori_img23 = double(imread([ori,'heart+salt&pepper.png']));


k = 2;
%用kmeans算法进行k类分割处理
new_img10 = Kmeans(ori_img10, k);
new_img20 = Kmeans(ori_img20, k);
new_img13 = Kmeans(ori_img13, k);
new_img23 = Kmeans(ori_img23, k);
```

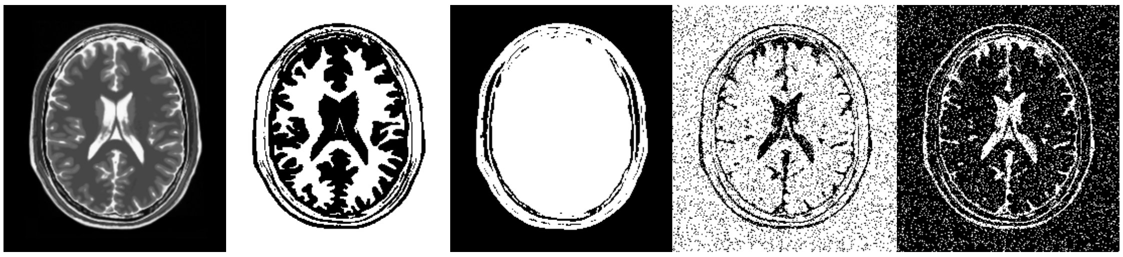
```
%保存处理结果
T = 'T2\';
imwrite(new_img10,[T,'K brain.png']);
imwrite(new_img20, [T,'K heart.png']);
imwrite(new_img13, [T,'K brain+salt&pepper.png']);
imwrite(new_img23, [T,'K heart+salt&pepper.png']);




function new_img = colouring(idx, r, c, k)
new_img_info = reshape(idx, r, c);
new_img = zeros(r, c, 3);
R1 = zeros(r, c);
G1 = R1;
B1 = R1;
switch k
    case 2
        % 用黑白两种颜色上色
        R1(new_img_info==1) = 0;
        G1(new_img_info==1) = 0;
        B1(new_img_info==1) = 0;
        R1(new_img_info==2) = 255;
        G1(new_img_info==2) = 255;
        B1(new_img_info==2) = 255;
    case 3
        % 用红绿蓝三种颜色上色
        R1(new_img_info==1) = 255;
        G1(new_img_info==1) = 0;
        B1(new_img_info==1) = 0;

        R1(new_img_info==2) = 0;
        G1(new_img_info==2) = 255;
        B1(new_img_info==2) = 0;

        R1(new_img_info==3) = 0;
        G1(new_img_info==3) = 0;
        B1(new_img_info==3) = 255;
    case 4
        % 用红绿蓝黄四种颜色上色
        R1(new_img_info==1) = 255;
        G1(new_img_info==1) = 0;
        B1(new_img_info==1) = 0;

        R1(new_img_info==2) = 0;
        G1(new_img_info==2) = 255;
        B1(new_img_info==2) = 0;

        R1(new_img_info==3) = 0;
        G1(new_img_info==3) = 0;
        B1(new_img_info==3) = 255;

        R1(new_img_info==4) = 255;
        G1(new_img_info==4) = 255;
        B1(new_img_info==4) = 0;
    otherwise
        error('please input k between 2 and 4')
end

new_img(:, :, 1) = R1;
new_img(:, :, 2) = G1;
new_img(:, :, 3) = B1;

end

function new_img = Kmeans(ori_img, k)
[r, c, ~] = size(ori_img);
mini = min(ori_img, [], 'all');
maxi = max(ori_img, [], 'all');
% 利用随机数初始化簇中心
centor0 = rand(k, 3) * 255;
centor1 = rand(k, 1) * (maxi-mini) + mini;

% 收敛阈值
err = 1e-12;
% 将RGB分量各转为kmeans数据格式
R = reshape(ori_img(:, :, 1), r*c, 1);
G = reshape(ori_img(:, :, 2), r*c, 1);
B = reshape(ori_img(:, :, 3), r*c, 1);
data = [R G B];
% idx记录每个点的聚类中心
idx = zeros(r*c, 1);

% 结果不收敛时继续循环
while(norm(centor1 - centor0, 'fro') >= err)
```

```
    centor0 = centor1;
    count = zeros(k, 1); % 每类点个数

    for i = 1 : r*c
        % 找到每个点对应的距离最近的聚类中心index
        [~, index] = min(sum((centor0 - data(i)).^2, 2));
        idx(i) = index;
        count(index) = count(index) + 1;
    end

    % 更新聚类中心
    for i = 1 : k
        centor1(i) = sum(data(idx==i), 1) / count(i);
    end
end

% 为分割的图像上色
new_img = colouring(idx, r, c, k);


end
```

```
clear
clc

%读取原图和部分噪声图
ori = 'Ori\';
ori_img10 = double(imread([ori,'brain.png']));
ori_img20 = double(imread([ori,'heart.png']));
ori_img13 = double(imread([ori,'brain+salt&pepper.png']));
ori_img23 = double(imread([ori,'heart+salt&pepper.png']));

%用otsu算法进行处理
new_img10 = otsu(ori_img10);
new_img20 = otsu(ori_img20);
new_img11 = otsu(ori_img13);
new_img21 = otsu(ori_img23);

%保存处理结果
ot = 'Otsu\';
imwrite(new_img10, [ot,'brain.png']);
imwrite(new_img20, [ot,'heart.png']);
imwrite(new_img11,[ot, 'brain+salt&pepper.png']);
imwrite(new_img21, [ot,'heart+salt&pepper.png']);
function new_img = otsu(ori_img)
[r, c] = size(ori_img);
%每个灰度值的直方图概率初始化
P = zeros(256, 1);
%累积概率初始化
sum_P = P;
%类中平均灰度值初始化
m = P;
%类间方差初始化
var = P;
for i = 1 : r
    for j = 1 : c
        P(ori_img(i, j)+1) = P(ori_img(i, j)+1) + 1;
    end
end
P = P / (r*c);
sum_P(1) = P(1);
for i = 2 : 256
    sum_P(i) = sum_P(i-1) + P(i);
    m(i) = m(i-1) + (i-1) * P(i);
end
mean = m(256);
for i = 1 : 256
    var(i) = (mean * sum_P(i) - m(i))^2 / (sum_P(i) * (1-
sum_P(i)));
end
[~, thre] = max(var);
threshold = ori_img > thre - 1;
new_img = zeros(r, c);
new_img(threshold) = 1;
new_img = new_img(1:r,1:c/3);
end
```

结果展示：

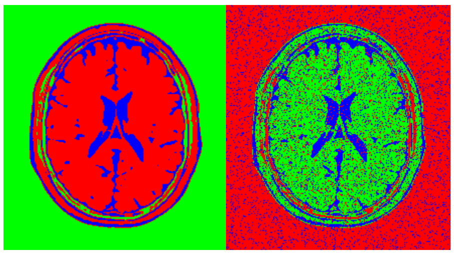
原图，Kmeans，global_Otsu，噪声-Kmeans，噪声-global-Otsu



三类分割：

结果展示：

Kmeans，噪声-Kmeans



讨论：

噪声图像的噪点灰度值会影响分割算法的权值。

解决方案：在进行分割前，先使用滤波器减小噪声。

[HW5-2] 请使用课程学习的形态学操作实现二值图像的补洞和离散点去除。形态学操作功能要自己代码实现，不能调用库。如下左图是有问题图像，右图是目标图像。

代码实现：

```
clear
clc

ori = 'Ori\';
ori_img = imread([ori,'zmic_fdu_noise.bmp']);

SE = zeros(5,5);

new_img = dila(ori_img,SE);
new_img = corr(new_img,SE);
imshow(new_img)
```

```
SE = zeros(5,5);
new_img = corr(new_img,SE);
new_img = dila(new_img,SE);
imshow(new_img)
imwrite(new_img, 'Seg\res.bmp');

function [new_img] = dila(ori_img, SE)
[r1,c1] = size(SE);
[r,c] = size(ori_img);
new_img = ones(r+r1-1,c+c1-1);
for i = 1 : r
    for j = 1 : c
        if ori_img(i,j) == 0
            new_img(i:i+r1-1,j:j+c1-1) = SE;
        end
    end
end
new_img = new_img(1+(r1-1)/2:r+1,1:c+(c1-1)/2);
end

function [new_img] = corr(ori_img,SE)
SE = 1-SE;
[r1,c1] = size(SE);
[r,c] = size(ori_img);
new_img = zeros(r+r1-1,c+c1-1);
for i = 1 : r
    for j = 1 : c
        if ori_img(i,j) == 1
            new_img(i:i+r1-1,j:j+c1-1) = SE;
        end
    end
end
new_img = new_img(1+(r1-1)/2:r+1,1:c+(c1-1)/2);
end
```

补洞：
先膨胀至无孔洞，再腐蚀回原始大小

离散点去除：
先腐蚀至无离散点，再膨胀回原始大小

结果呈现：

以下以此为，原图，噪声图，处理后图片