可视化作业5

`18300290007 加兴华`

`注：本次作业均在matlab环境下完成`

[HW5-1]

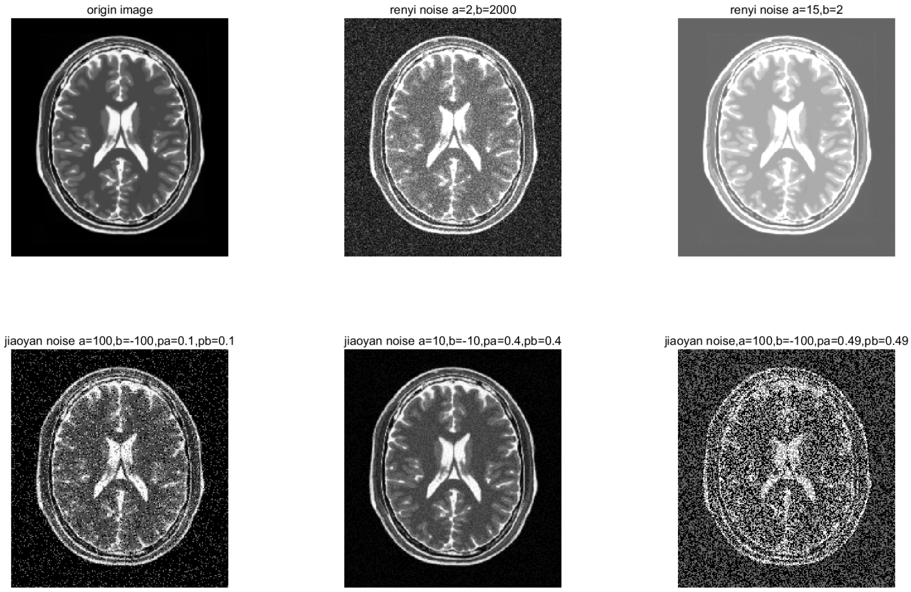
针对对[大脑](#)、[心脏图像](#)（或其他多类图像），生成其对应不同类型（瑞利噪声、椒盐噪声）的不同强度（至少2种高强度）的噪声污染图像。

用于增加噪声的函数如下：

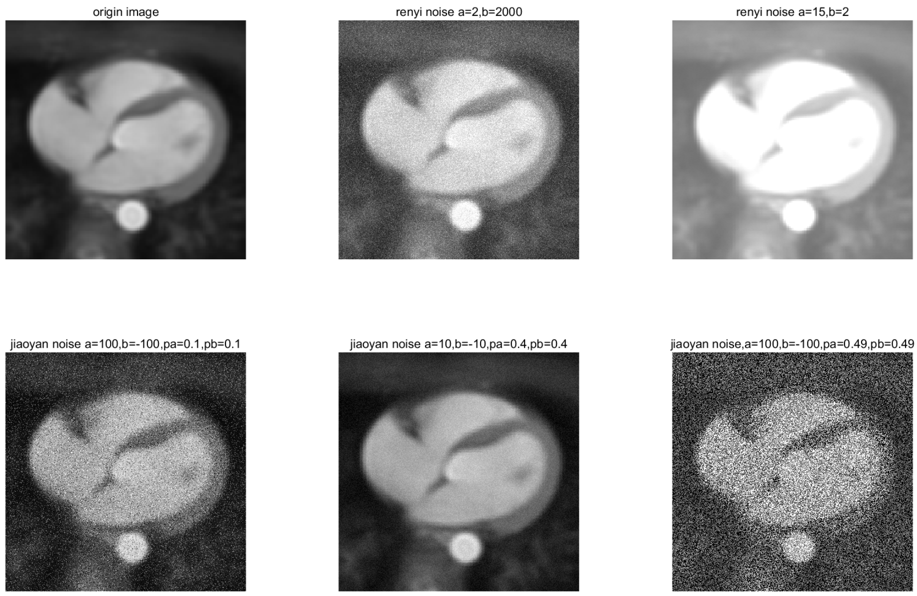
```matlab
% 瑞利噪声
function x=renyi(img,a,b)
[m,n]=size(img);
noise=rand(m,n); % 生成01均匀分布的同规模矩阵
% 将每个元素按噪声pdf反变换等效按噪声分布生成噪声矩阵
noise=a+(-b.*log(1-noise)).^0.5;
% 叠加
x=img+uint8(noise);
end

% 椒盐噪声
function y=jiaoyan(img,a,b,pa,pb)
[m,n]=size(img);
noise=zeros(m,n); % 申请噪声矩阵空间
cdf=rand(m,n); % 生成01均匀分布的同规模矩阵
% 将每个元素按噪声pdf反变换等效按噪声分布生成噪声矩阵
noise(cdf<pa)=a;
noise(cdf>1-pb)=b;
% 叠加
y=uint8(double(img)+noise);
end
```

应用在'大脑图像'上可获得如下结果：



应用在'心脏图像'上可获得如下结果：



分析：

【1】瑞利噪声，噪声点的多寡取决于b的大小，b越大噪声点越多；而a并非带来直观上的噪声，a会产生'灰度'意义上的噪声：使原图的亮度发生偏移。

【2】椒盐噪声，a值决定白噪声的亮度，b值决定黑噪声的亮度；pa决定白噪声的密度，pb决定黑噪声的密度。可以观察到，a和b的值越大，噪声就越明显；pa和pb的值越大（加起来不能超过1），噪声点就越多；另外，pa与pb相等时，值小时白噪音更明显，值大时黑噪音更明显，这可能是因为人眼视觉特性导致的。

[HW5-2]

实现OSTU二类分割算法，K类均值分类的分割算法和基于高斯混合模型的分割算法，并测试一下上述高强度噪声污染的图像的分割结果：

（1）

测试二类分割，并对比三个算法结果的差异；

otsu函数实现：

```
function a=otsu(img)
% 统计图像信息到bin
[m,n]=size(img);
bin=zeros(256,1);
for i=1:m
    for j=1:n
        bin(img(i,j)+1)=bin(img(i,j)+1)+1;
    end
end
% 初始化
s_max =[0,0];
a=zeros(m,n);
N=m*n;  % 像素个数
i=0:255;    %递增行向量，用于期望的向量表示
for threshold=1:256
    u=0;
    n_0 = sum(bin(1:threshold));  % 阈值以下像素数
    n_1 = sum(bin(threshold:256));  % 阈值以上像素数
    % 两侧频率
    w_0 = n_0/N;
    w_1 = n_1/N;
    % 阈值下平均灰度
    if(n_0>0)
        u_0 = i(1:threshold)*bin(1:threshold)/n_0;
    else
        u_0=0;    %考虑极端划分
    end
    % 阈值上平均灰度
    if(n_1>0)
        u_1 = i(threshold:256)*bin(threshold:256)/n_1;
    else
        u_1=0;    %考虑极端划分
    end
    % 总平均灰度
    u = w_0*u_0 + w_1*u_1;
    % 类间方差
    Dbet2 = w_0*((u_0-u)^2) + w_1*((u_1-u)^2);
    % 跟先前最优比较(取绝对大者意味着右偏，二分时阈值需分配给左半边)
    if (Dbet2>s_max(2))
        s_max=[threshold-1,Dbet2];
    end
end
t=s_max(1);
for i=1:m
    for j=1:n
        if img(i,j)>t a(i,j)=255; end
    end
```

```
end
end
```

k—means函数实现:

```
function b=mykmeans(img,k)
%-------------
% 随机初始化
u=rand(k,1)*255;
u=sort(u);
img=double(img); % 浮点化防止后面运算报错
[m,n]=size(img);
c=zeros(m,n); % 每个像素点的类记录矩阵
I=ones(m,n);  % 用于后续便捷表达的辅助矩阵
d=ones(k,1);  % 每一类的迭代差距
err=10;       % 初始差距
%--------------
%正式迭代
while(err>0.1)
%根据当前每类的中心界定像素点类型
    for i=1:m
        for j=1:n
            min=255;
            for r=1:k
                if abs(img(i,j)-u(r))<=min
                    min=abs(img(i,j)-u(r));
                    c(i,j)=r-1;
                end
            end
        end
    end
%计算每一类点的灰度期望作为下一迭代的类中心
    for r=1:k
        temp=u(r);
        tot=sum(sum(img(c==r-1)));
        con=sum(sum(I(c==r-1)));
        u(r)=tot/con;
        d(r)=abs(u(r)-temp);
    end
%以最大迭代距离作为本次误差
    err=max(d);
end
%转换成灰度图数据格式
b=uint8(c/(k-1)*255);
end
```

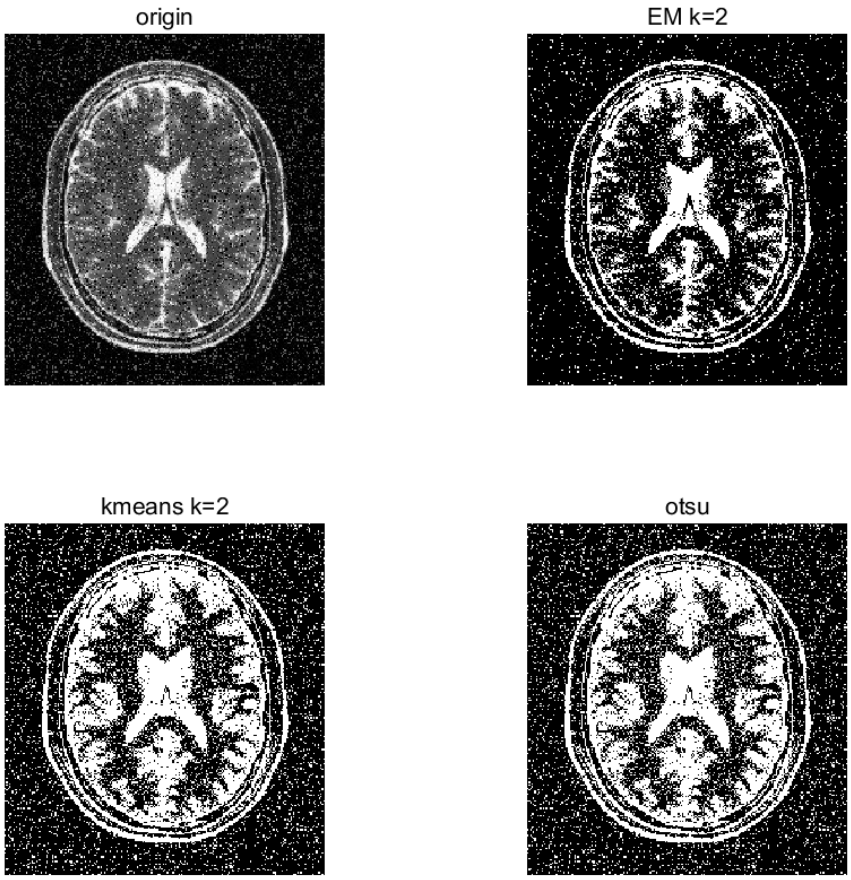
EM法函数实现:

```
function c=em(img,k)
% double化防止出错
img=double(img);
% 设置初始均值和方差，均值为随机在各色块中选点并取整所得
mu=rand(k,1)*255;
```

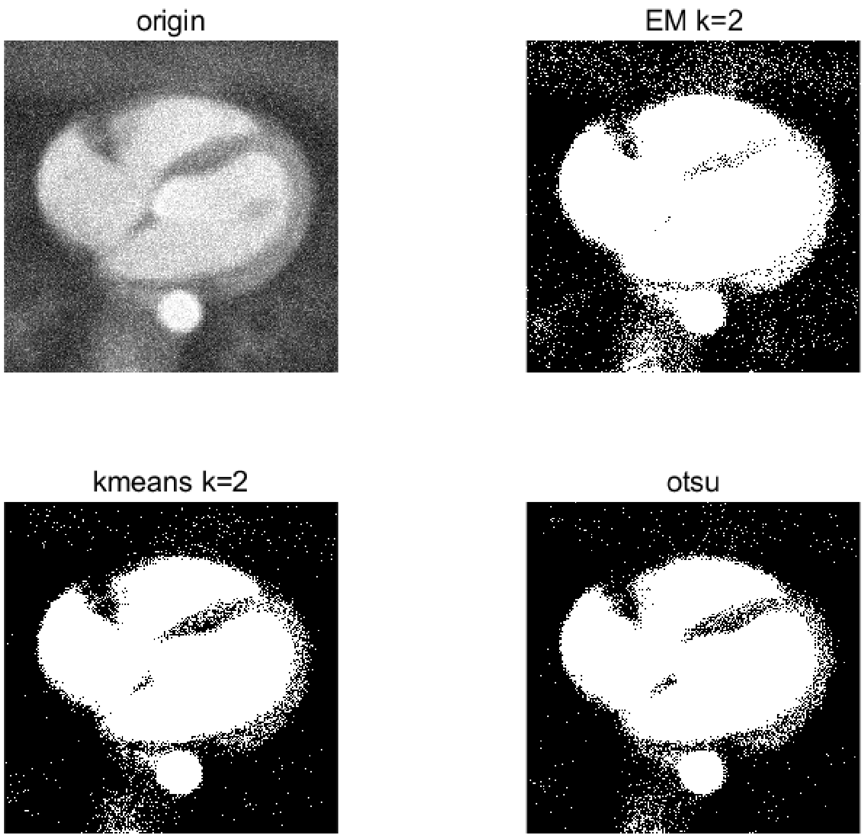
```
mu=sort(mu);
sigma =100*ones(1,k);
[m,n]=size(img);
px = zeros(m,n,k);
%全局每个点为某一类的概率
mypi = rand(1,k);
mypi = mypi/sum(mypi);%将类概率归一化
%以迭代次数来作为停止的条件
stopiter = 20;
iter = 1;
while iter <= stopiter
    %----------E-------------------
    N=zeros(m,n,k);
    for i=1:k
        for x=1:m
            for y=1:n
                N(x,y,i)=normpdf(img(x,y),mu(i),sigma(i))*mypi(i);
            end
        end
    end
    % 防止分母为0 生成一个无0近似阵
    N2=N;
    N2(N<0.00000001)=0.000000001;
    for x=1:m
        for y=1:n
            sumk_N=sum(N2(x,y,:));
            px(x,y,:)=N(x,y,:)/sumk_N;
        end
    end
    %----------M---------------------
    %更新参数集
    sum_all_px=sum(sum(sum(px)));
    for i=1:k
        sum_xy_px=sum(sum(px(:,:,i)));
        mu(i)=sum(sum(px(:,:,i).*img))/sum_xy_px;
        sigma(i) =sqrt(sum(sum(px(:,:,i).*((img-mu(i)).^2)))/sum_xy_px);
        mypi(i)=sum_xy_px/sum_all_px;
    end
    iter=iter+1;
end
c=zeros(m,n);
for x=1:m
    for y=1:n
    % 返回x，y处最大概率类型下标
        [~,c(x,y)]=max(px(x,y,:));
    end
end
c=uint8((c-1)/(k-1)*255);
end
```

首先需要说明，由于EM法我的初始均值、方差、初始类概率均为随机生成，且迭代次数设置不多，因此EM法可能会有较大的波动性。

对于椒盐噪声，EM法的最佳还原效果比另两者好，另外两种方法差异不大：



而对于瑞利噪声，经过多次运行代码后，都观察到EM法总是还原效果最差，而otsu竟然此时处理效果最好：



（2）

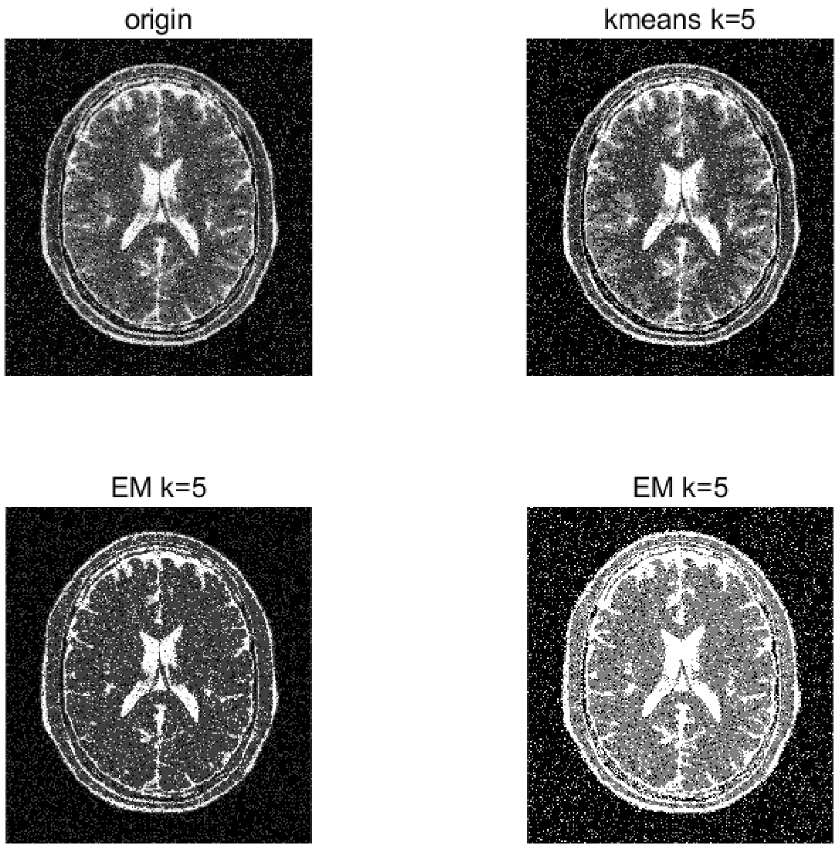
测试多类（大于等于三类）分割（请自己设定分割标签类别的个数），并对K类均值分类的分割算法和基于高斯混合模型的分割结果的差异;

将EM法初始均值设置为kmeans中心:

```
%修改成以kmeans中心作为初始均值
function c=em(img,k)
img=double(img);
% 设置初始均值和方差，均值为随机在各色块中选点并取整所得
[mu,~]=mykmeans(img,k);
sigma =100*ones(1,k);
[m,n]=size(img);
px = zeros(m,n,k);
%全局每个点为某一类的概率
mypi = rand(1,k);
mypi = mypi/sum(mypi);%将类概率归一化
%以迭代次数来作为停止的条件
stopiter = 20;
iter = 1;
while iter <= stopiter
    %----------E-------------------
    N=zeros(m,n,k);
    for i=1:k
        for x=1:m
            for y=1:n
                N(x,y,i)=normpdf(img(x,y),mu(i),sigma(i))*mypi(i);
```

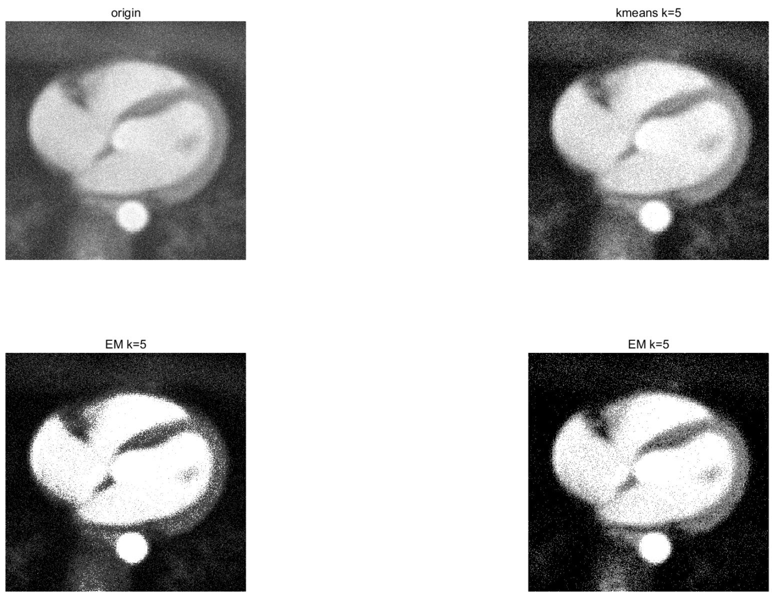
```
                end
            end
        end
        N2=N;
        N2(N<0.00000001)=0.000000001;% 防止分母为0 生成一个无0近似阵
        for x=1:m
            for y=1:n
                sumk_N=sum(N2(x,y,:));
                px(x,y,:)=N(x,y,:)/sumk_N;
            end
        end
        %----------M--------------------
        %更新参数集
        sum_all_px=sum(sum(sum(px)));
        for i=1:k
            sum_xy_px=sum(sum(px(:,:,i)));
            mu(i)=sum(sum(px(:,:,i).*img))/sum_xy_px;
            sigma(i) =sqrt(sum(sum(px(:,:,i).*((img-mu(i)).^2)))/sum_xy_px);
            mypi(i)=sum_xy_px/sum_all_px;
        end
        iter=iter+1;
    end
    c=zeros(m,n);
    for x=1:m
        for y=1:n
    % 返回(x, y)处最大概率类型给c
            [~,c(x,y)]=max(px(x,y,:));
        end
    end
    c=uint8((c-1)/(k-1)*255);
    end
```

一次运行获得下图:



这说明，图片带椒盐噪声时，EM法的最优还原效果确实会比kmeans好，但每次运行不必然达到最优，有时甚至效果会劣化。

下面再看看面对瑞利噪声会发生什么：



观察到EM法分级的对比度普遍高于kmeans，虽然细看噪声点也随之更明显，但同时原图的轮廓也更加好辩认，有利有弊。

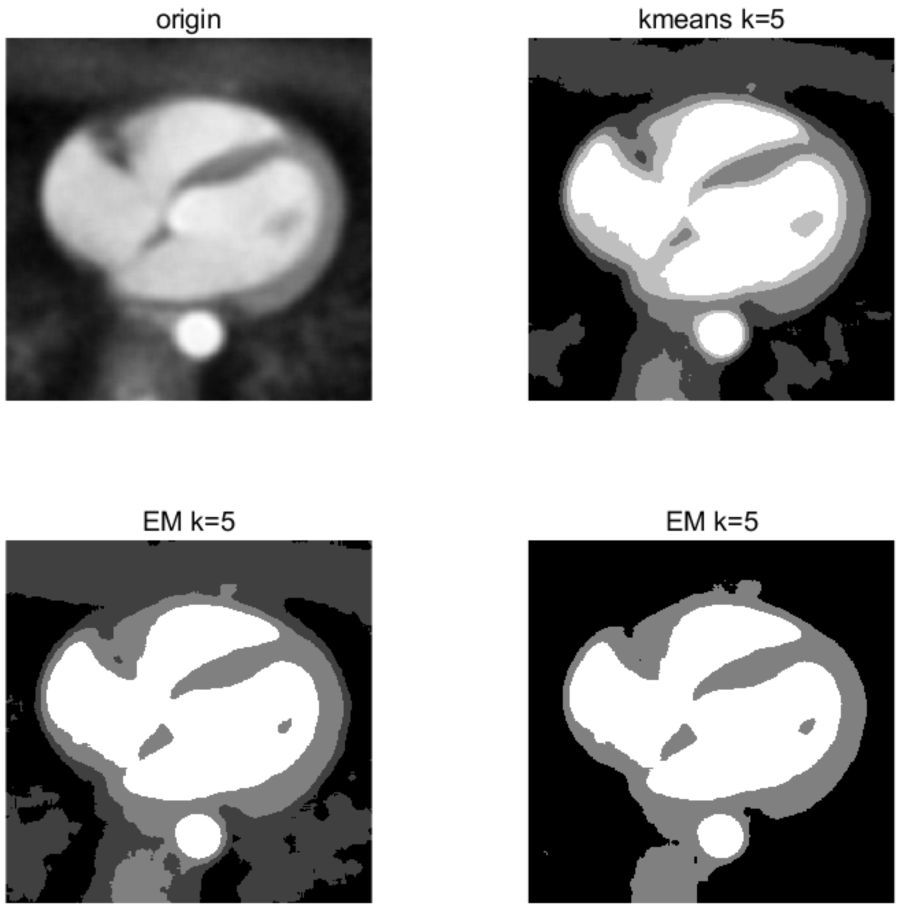
（3）

针对噪声图像，讨论为什么分割的结果不准确。用什么方法可以取得更好的分割结果。

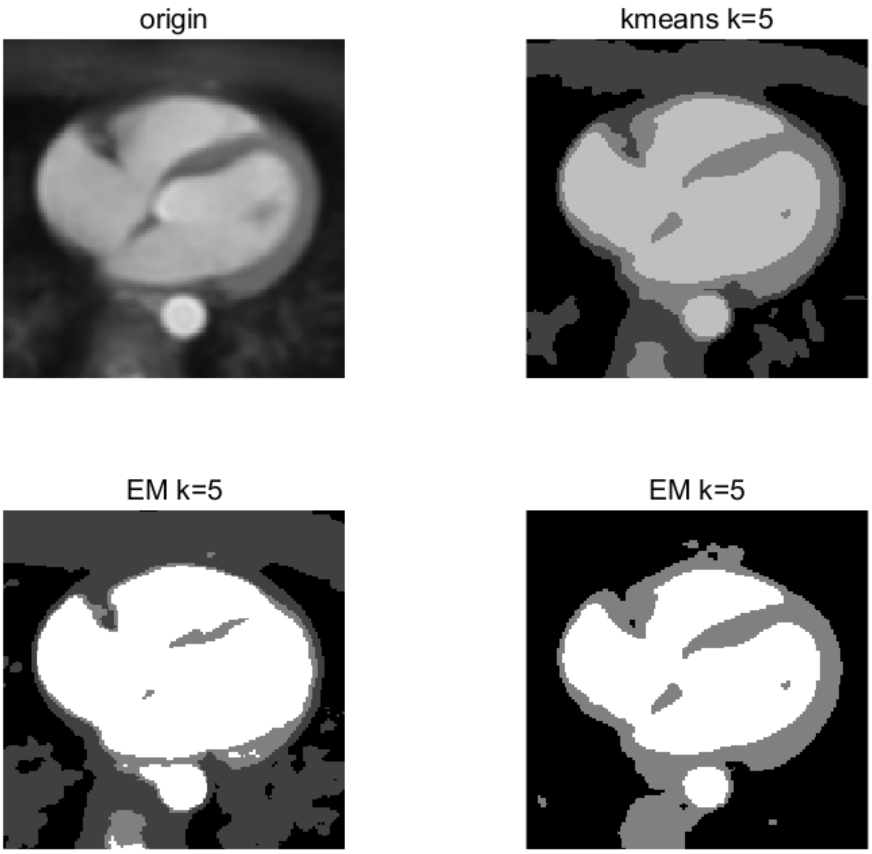
答：

分割结果不准确其实就是图像分割操作不能很好的识别出噪声点。仔细分析，可以把噪声点看成是一阶导突增的位置，如果我们能够减缓这种突增，就能使噪声点接近未污染前状态，一方面渐小噪声图与原图聚类时的差异，另一方面也能使噪声点有更高的概率被划分到为污染前的类别。

基于此，下面我将噪音图像做了平滑处理，再进行分割，结果如下：



而直接对无污染原图分割结果如下：



能观察到前后的右下方图几乎完全一致！说明平滑处理确实是一种有效的处理噪音的方式。