中 国 地 质 大 学

本科生课程报告封面

课 程 名 称 数字图像处理与分析

教 师 姓 名 陈伟涛

本科生姓名 田冰

本科生学号 20181001910

本科生专业 计算机科学与技术

所 在 院 系 计算机学院

日 期 2020.11.04

目录

一、实验目的

理解经典的区域生长算法过程，了解经典的区域生长在图像分割中的优劣，以及对该方法进行改进和创新。

二、实验内容

对灰度图像："jx_3.jpg"分别实现 2 种图像分割方法：

1.经典区域生长方法；

2.改进的区域生长方法；

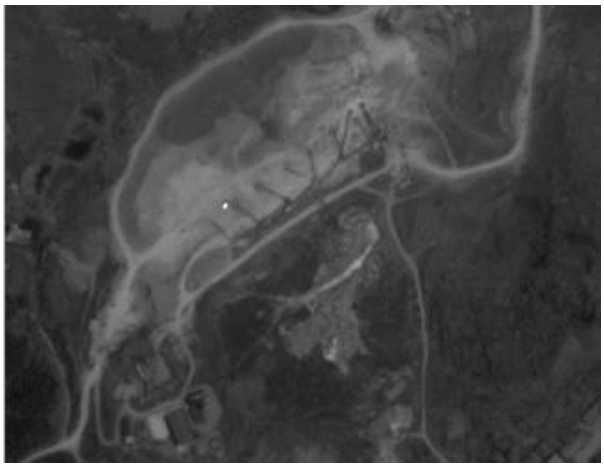

Figure 1 原图

改进的区域生长方法是基于经典区域生长方法完成的。具体要求如下：

（1）首先，基于经典区域生长方法，实现图像分割；

（2）然后，基于经典区域生长方法，自己查阅文献，设计一种改进的区域生长算法，实现图像分割；

（3）目视比较分析 2 种方法的分割效果，概括改进方法分割效果的优点和不足之处。

三、算法设计&主要代码

区域生长算法相当于从下到上对像素进行合并，从满足检测点（或手动选择像素点）开始，在各个方向上把具有相似性质的像素集合起来构成一个区域。

应用区域生长需要解决的问题：

（1）如何选择或确定一组能代表所在区域的种子像素

（2）怎样确定在生长过程中将相邻像素包括进来的准则

（3）如何制定让生长过程停止的条件或规则

3.1 经典的区域生长算法：

3.1.1 算法设计：

经典的区域生长算法是通过随机或指定一组区域作为种子像素，根据图片确定种子区域灰度值的平均值与邻域像素点灰度值之差的最大值（阈值），灰度值之差小于阈值作为相邻像素被包括进来的准则，没有邻域像素点满足准则的时候作为停止生长的条件。

具体过程即，先根据经验确定生长过程中相邻像素点灰度值之差的最大值（阈值），将种子像素点周围邻域中与种子像素灰度值之差小于阈值的像素合并到种子像素所在区域中。将这些新的像素继续当作种子像素进行上面操作过程，直到再没有满足条件的像素可以被包括进来，最后的区域即生长成的区域。

3.1.2 主要代码：

```
while count>0                    %判断是否继续生长
 s=0;                            %记录一点周围的灰度值之和
 count=0;                        %记录新加入的像素点
 for i=1:M
     for j=1:N
       if J(i,j)==1
        if (i-1)>0 && (i+1)<(M+1) && (j-1)>0 && (j+1)<(N+1)        %检测边界点
         for u= -1:1
          for v= -1:1  %设置偏移量
            if  J(i+u,j+v)==0 && abs(I(i+u,j+v)-seed_point)<=threshold
                %判断是否未存在于输出矩阵 J，并且为符合阈值条件的点
               J(i+u,j+v)=1; %设置为白点
               count=count+1;
               s=s+I(i+u,j+v);
            end
          end
         end
        end
       end
     end
 end
 count_sum=count_sum+count;
 gray_sum=gray_sum+s;
 seed_point=gray_sum/count_sum; %计算新区域的灰度平均值
end
```

3.1.3 主要代码解析：

首先设置阈值 threshold,每次循环检查是否还有未生长的区域像素点，若有则开始生长。设置区域像素点总数变量 count_sum 和区域像素点总灰度值变量 gray_sum，以便用来计算区域内像素灰度的平均值 seed_point。每个像素点在生长的过程中首先检查是否为边界像素点（因为边界像素点无法使用偏移量操作邻域像素点），若不是，则检查该生长的像素点的邻域是否有满足生长准则的点（abs(I(i+u,j+v)-seed_point)<=threshold）,若满足准则，则将满足准则的点加入区域内，并记录数量 count 和灰度值 s，生长结束更新区域内的灰度平均值。

3.2.改进的区域生长算法：

3.2.1 思路由来：

由应用区域生长需要解决的三个问题中看，要想对经典的区域生长进行改进，可以分别从这三个问题点出发—**初始种子点选择**、**生长准则**、**停止生长条件**，并且初始种子点选择和生长准则方面有很大的改进空间。

在查阅相关的资料和论文之后，笔者发现：1.初始种子点的选择对最终的图像分割的结果有很大的影响，所以可以从初始种子点入手，在开始生长前选择出对分割结果最有利的种子点；2.显而易见，单是经典区域生长方法中设置的阈值不同，就会对图像分割结果有很大的影响，所以还可以从阈值的选择或者生长准则的优化方面入手，找出对图像分割最优的阈值或者比经典区域生长准则更优的生长准则。

根据手中查到的资料和个人偏好，笔者从制定更优的生长准则来改进算法，并且经过实验的对比分析，一种基于自适应的区域生长算法比经典的区域生长算法在图像分割效果上表现得更加出色！下面是详细介绍。

基于自适应的区域生长算法：

3.2.2 区域的定义：

算法定义两个区域Ω(x)和Ωin(x)，其中Ω(x)为正在生长点的邻域，Ωin(x)表示正在生长点的邻域与已生长区域 Rt 的交集，公式表示为(δ为距离半径，即 x 邻域的大小)：

$$\Omega(x) = \{y \in Rt \cup \overline{Rt} | d(x,y) < \delta\}$$

$$\Omega in(x) = \{y \in Rt | d(x,y) < \delta\}$$

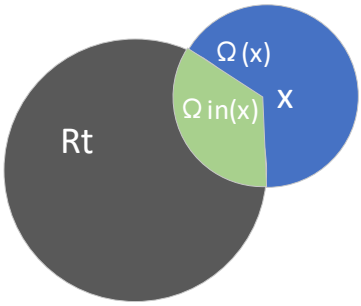

Figure 2 区域示意图

3.2.3 局部参数：

自适应区域生长算法主要基于两个局部参数：局部区域的平均灰度$\mu^{[t]}{}_x$和局部区域的平均梯度$||\bar{\nabla}||^{[t]}{}_x$，这两个参数的计算主要依赖于像素点 x、迭代次数 t、以及相关的局部区域。

在经典的区域生长算法中，使用全局的平均灰度值，即目前已经生长到的整个区域均值，也就是 Rt 区域平均灰度值，然而，自适应区域生长算法中所用到的平均灰度值仅是通过计算那些与像素 x 相邻，且在 Rt 区域内的像素灰度均值，即 Ω in(x)区域的平均灰度值，其局部区域的平均灰度计算公式如下：

$$\mu^{[t]}{}_x = \frac{1}{card(\Omega in(x))} \times \sum_{y \in \Omega in(x)} I(y)$$

其中，card(Ω in(x))表示 Ω in(x)区域像素个数，I(y)表示像素 y 点的灰度值。

局部区域平均梯度$||\bar{\nabla}||^{[t]}{}_x$是指 Ω (x)区域内所有像素的平均梯度，其计算公式如下：

$$||\bar{\nabla}||^{[t]}{}_x = \frac{1}{card\Omega(x)} \times \sum_{y \in \Omega(x)} ||\nabla I(y)||$$

其中，card Ω (x)表示 Ω (x)区域内像素个数，$\nabla I(y)$表示像素 y 点的梯度值。

3.2.4 生长准则：

自适应区域生长的相似性判断准则是基于局部区域的平均灰度$\mu^{[t]}{}_x$和局部区域的平均梯度$||\bar{\nabla}||^{[t]}{}_x$而设定的，其准则表达如下：

$$\mu_x^{[t]} - \alpha \cdot \|\bar{\nabla}\|_x^{[t]} < I(y) < \mu_x^{[t]} + \beta \cdot \|\bar{\nabla}\|_x^{[t]}$$

其中，I(y)表示像素点 y 的灰度值，α 和 β 为可人工调节的常量系数。从公式可以看出，相似性判断准则完全是自适应的，允许像素点 y 的灰度值与局部均值之间有一定变化的强度，而这个变化强度是基于局部梯度的一个函数，如果变化超出了这个范围，则像素点 y 就会被认为是异常值，即该像素不属于同类区域。

3.2.5 种子点的选取：

在经典的区域生长算法中，用户在选择单个初始种子点时存在误选的情况，选中的可能是噪声点，因此，本算法采用种子点的 3X3 邻域作为初始种子区域，这样可以适当避免种子点的误选和噪声的影响。

3.2.6 算法实现流程图：

自适应区域生长算法实现流程图

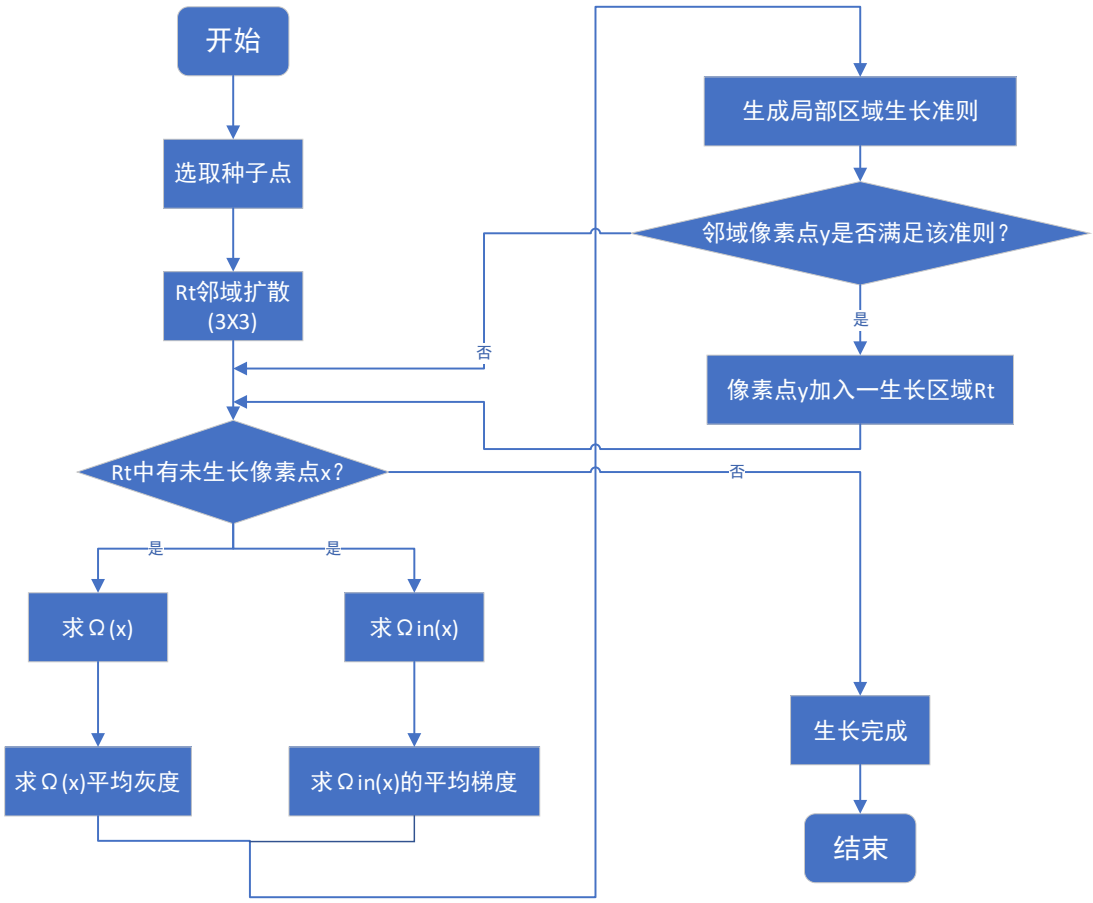

Figure 3 算法流程图

3.2.7 主要代码：

```
1.  gausFilter = fspecial('gaussian',[3 3],0.5);
2.  I = imfilter(I,gausFilter,'replicate');
3.  [DX,DY]=gradient(I);  %计算图像两个方向上的梯度
4.  G=sqrt(DX.^2+DY.^2);   %计算梯度矩阵
```

代码解释：先对图像进行高斯去噪，使用 matlab 工具箱 gradient 函数对图像求梯度，计算每个像素点的平均梯度，这一步便于生长过程中计算局部参数

```
1.  while count>0
2.      s=0;      %Ωin(x)像素灰度值和
```

```
    gra=0;        %Ω(x)像素梯度和
    count=0;      %待生长像素点数量
    count_ux=0;    %Ωin(x)像素数量
    count_ugra=0;   %Ω(x)像素数量
    for i=1:M
        for j=1:N
            if (i-1)>0 && (i+1)<(M+1) && (j-1)>0 && (j+1)<(N+1)
                for u=-1:1   %第一次遍历邻域
                    for v=-1:1
                        gra=gra+G(i+u,j+v);
                        count_ugra=count_ugra+1;
                        if J(i+u,j+v)==1
                            s=s+I(i+u,j+v);
                            count_ux=count_ux+1;
                        end
                    end
                end
                ux=s/count_ux;  %计算局部区域的平均灰度
                ugra=gra/count_ugra;  %计算局部区域的平均梯度
                for u=-1:1
                    for v=-1:1
                        if J(i+u,j+v)==0 && I(i+u,j+v) > (ux-a*ugra) && I(i+u,j+v) < (ux+b*ugra)  %相似性准则
                            J(i+u,j+v)=1;
                            count=count+1;
                        end
                    end
                end
            end
        end
    end
end
```

3.2.8 代码解释：

在循环生长的过程中，每次生长都进行两次像素邻域点的遍历。

第一次遍历求出邻域中相对于已生长区域 Rt 的 Ω(x)和 Ω in(x)区域，并记录相应区域的像素点数、像素点灰度值、像素点梯度值，遍历结束后，根据记录的数据计算相应的局部参数$\mu^{[t]}{}_x$和$||\overline{\nabla}||^{[t]}{}_x$。

第二次遍历根据之前求出的局部参数设计该邻域点对应的相似性准则 I(i+u,j+v) > (ux-a*ugra) && I(i+u,j+v) < (ux+b*ugra)，其中变量 a 和 b 可以根据图片分割具体情况进行调整，基于此次图片的分割，设置 a=1.5 b=0.5。将满足该准则的像素点加入到已生长区域 Rt 中，并记录加入的点，用于下一次生长。

局部参数在每次循环中都会发生改变(ux;ugra)，对应的相似性准则也会发生改变，这就是自适应区域生长算法的核心，类似于“分而治之，因地制宜”的思想，可以有效地避免经典区域生长对图像过度分割的问题。

四、结果对比分析

Result 1 经典区域生长结果表

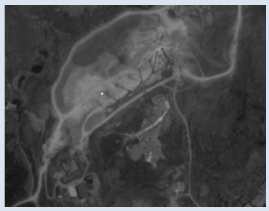

Figure 4 原图

Figure 5 threshold=0.1

Figure 6 threshold=0.065

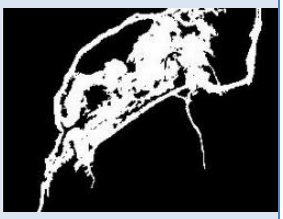

Figure 7 threshold=0.08

Result 2 自适应区域生长结果表

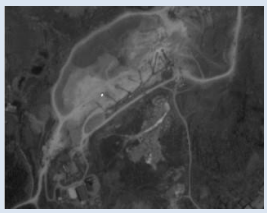

Figure 8 原图

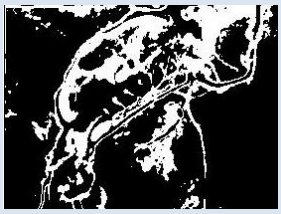

Figure 9 a=2 b=1.2

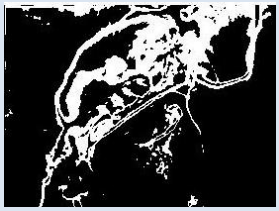

Figure 10 a=1.5 b=1.2

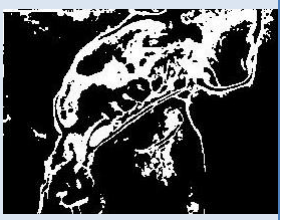

Figure 11 a=1.5 b=0.5

Result 3 经典 VS 自适应 对比图

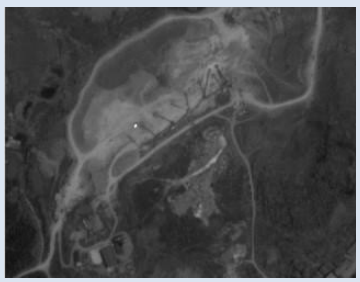

Figure 12 原图

Figure 13 经典区域生长分割图

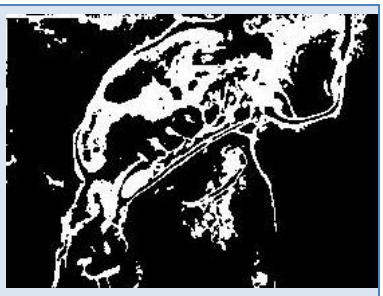

Figure 14 自适应区域生长分割图

Figure 15 原图

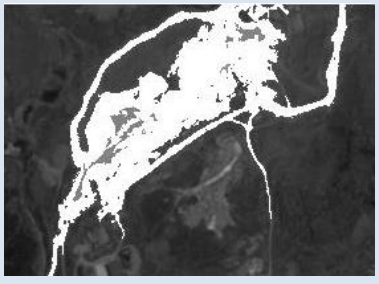

Figure 16 经典区域生长分割融合图

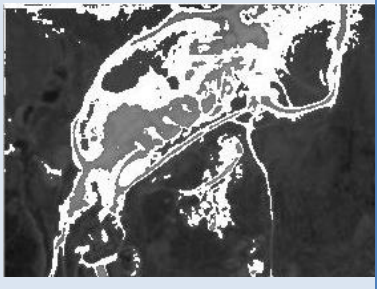

Figure 17 自适应区域生长分割融合图

Result 4 经典 VS 自适应 空洞对比

Figure 18 经典区域生长分割空洞图

Figure 19 自适应区域生长分割空洞图

Result 5 经典 VS 自适应 总体对比

分割方法	平均时间/s	分割效果
经典区域生长	0.05	过度分割、空洞较多
自适应区域生长	0.19	分割清楚、空洞较少

4.1 实验结果分析：

两算法对图像的分割效果都受参数的影响（阈值、a&b），其中经典的区域生长算法在阈值 threshold 为 0.08 的情况下，效果最好；自适应区域生长算法在 a=1.5 b=0.5 的情况下，效果最好。但是在相同情况下，基于局部平均灰度和局部平均梯度这两个参数，对区域生长算法中的相似性准则进行改进，相比较经典的区域生长算法，对如图所示卫星图的分割效果更好，可以使图像中的细节分割开来，并且在相同的条件下，分割得到的图像空洞更少。

五、收获与体会

首先，通过这次实验，我真正意义上实现了一次论文算法的代码实现，虽然所查阅的论文为中文论文且算法不是很复杂，却大大提高了我的自信心，以前总是觉得论文上面的算法包含大量的数学公式，用代码实现很困难，但是这次我做到了！

其次，我熟悉了区域生长算法。从经典的区域生长到本文所讲述的自适应区域生长算法，虽然前期没能靠自己来编程实现，但是学习了网上的资料之后，我就对该算法有了大致的编程思路，并且能进行独立的编程实现，算法实现的过程中又加深了我对该算法的理解，实现该算法的方式有很多种，我使用了一种比较简单的方式来实现，后面还可以使用集合的思想来实现。

最后，在查阅论文的过程中，我印象最深的就是图像分割技术主要应用于医学领域，比如 CT 图像背景分割、大脑轮廓提取等等。我相信图像处理和图像分割的进一步发展，会极大地推动人类医学的进步!前文提到对经典区域生长算法

进行改进可以从种子点的选取和生长准则等方面入手，本文所述算法只是从生长准则入手对算法进行了改进，后面的时间我会查询资料了解怎样选取对图像分割结果有利的初始种子点，相信也能对结果有很大的提升。

数字图像与处理这门课我收获很多，老师上课会根据课内知识补充其他的知识点，让我对图像有了一个全新的认识！

六、实验源代码

6.1 经典区域生长算法：

```
I = imread('jx.png');
I = rgb2gray(I);%转化为单通道灰度图
I = im2double(I); %图像灰度值归一化到[0,1]之间
gausFilter = fspecial('gaussian',[3 3],0.5);
I = imfilter(I,gausFilter,'replicate');
[M,N]=size(I);
J=zeros(M,N);                %定义输出图像矩阵，初始为零矩阵
figure,imshow(I,[]);title('原始图像');
hold on;
[y,x] = getpts;%鼠标取点   回车确定
tic;
xr = round(x(1));%选择种子点
yr = round(y(1));
seed_point=I(xr,yr);
J(xr,yr)=1;                  %将 J 中与所取点相对应位置的点设置为白点
gray_sum=seed_point;                %储存符合区域生长条件的点的灰度值的总和
count_sum=1;                 %储存符合区域生长条件的点的总个数
count=1;                     %每次判断一点周围八点符合条件的新点的数目
threshold=0.08;               %阈值
while count>0                %判断是否有新的符合生长条件的点，若没有，则结束
 s=0;                        %记录判断一点周围八点时，符合条件的新点的灰度值之和
count=0;
 for i=1:M
     for j=1:N
       if J(i,j)==1
        if (i-1)>0 && (i+1)<(M+1) && (j-1)>0 && (j+1)<(N+1)
        %判断此点是否为图像边界上的点
         for u= -1:1                          %判断点周围八点是否符合阈值条件
          for v= -1:1                         %u,v 为偏移量
            if  J(i+u,j+v)==0 && abs(I(i+u,j+v)-seed_point)<=threshold
                              %判断是否未存在于输出矩阵 J，并且为符合阈值条件的点
               J(i+u,j+v)=1;
                             %将符合以上两条件的点在 J 中与之位置对应的点设置为白点
```

```
                count=count+1;
                s=s+I(i+u,j+v);
              end
            end
          end
        end
      end
    end
  end
  count_sum=count_sum+count;
  gray_sum=gray_sum+s;
  seed_point=gray_sum/count_sum;      %计算新的灰度平均值
end
toc;
figure,imshow(J);
title('分割后图像');
figure,imshow(J+I);
title('分割后图像');
```

6.2 基于自适应的区域生长算法：

```
I = imread('jx.png');
I = rgb2gray(I);%转化为单通道灰度图
I = im2double(I); %图像灰度值归一化到[0,1]之间
gausFilter = fspecial('gaussian',[3 3],0.5);
I = imfilter(I,gausFilter,'replicate');
[M,N]=size(I);
[DX,DY]=gradient(I);  %计算图像两个方向上的梯度
G=sqrt(DX.^2+DY.^2);   %计算梯度矩阵
J = zeros(size(I)); % 主函数的返回值，记录区域生长所得到的区域
figure,imshow(I,[]);
hold on;
[y,x] = getpts;%鼠标取点   回车确定
tic;
xr = round(x(1));%选择种子点
yr = round(y(1));
J(xr,yr)=1;                %将 J 中与所取点相对应位置的点设置为白点
count=1;                   %每次判断一点周围八点符合条件的新点的数目
a=1.5;   %自适应生长的范围
b=0.5;
for u=-1:1
    for v=-1:1
        J(xr+u,yr+v)=1;
    end
```

```
end
while count>0
    s=0;     %Ωin(x)像素灰度值和
    gra=0;       %Ω(x)像素梯度和
    count=0;      %待生长像素点数量
    count_ux=0;    %Ωin(x)像素数量
    count_ugra=0;   %Ω(x)像素数量
    for i=1:M
        for j=1:N
            if (i-1)>0 && (i+1)<(M+1) && (j-1)>0 && (j+1)<(N+1)
                for u=-1:1   %第一次遍历邻域
                    for v=-1:1
                        gra=gra+G(i+u,j+v);
                        count_ugra=count_ugra+1;
                        if J(i+u,j+v)==1
                            s=s+I(i+u,j+v);
                            count_ux=count_ux+1;
                        end
                    end
                end
                ux=s/count_ux;  %计算局部区域的平均灰度
                ugra=gra/count_ugra;  %计算局部区域的平均梯度
                for u=-1:1
                    for v=-1:1
                        if J(i+u,j+v)==0 && I(i+u,j+v) > (ux-a*ugra) && I(i+u,j+v) < (ux+b*ugra)  %相似性准则
                            J(i+u,j+v)=1;
                            count=count+1;
                        end
                    end
                end
            end
        end
    end
end
toc;
figure,imshow(J);
title('分割后图像');
figure,imshow(J+I);
title('分割后图像');
```

参考文献

[1]彭丰平,鲍苏苏,曾碧卿,基于自适应区域生长算法的肝脏分析[计算机工程与应用].2010

[2]肖明尧,李雄飞,张小利,张刘,基于多尺度的区域生长的图像分割算法[吉林大学学报(工学报)].2017.09

[3]张乐,项安,区域生长算法的改进及其在异物检测中的应用[同济大学]2018.04

[4]魏津瑜,施鹤南,苏思沁,基于改进算法的自动种子区域生长图像分割[中南大学学报(自然科学版)]2013.S2

[5]李仔麒,马慧彬,李殿奎,范蕊,改进区域生长法的肝部 CT 图像 ROI 提取[计算机技术与发展]2019.01