# 实验报告

# 颗粒图像的分割及计数系统

姓名： 王罗州

学号：41610074

# 颗粒物分割及计数系统内容简介：

颗粒物分割及计数及根据提供的含有大量颗粒物的图片，首先使用一系列图像算法对图像处理，然后再用判定算法，对图像中的颗粒物进行统计，获得每一个颗粒物的位置，周长和面积，并且统计出颗粒物的个数。

本系统采用Matlab进行编程，并且使用Matlab编写前端GUI，前端获取图片，后台计算并统计每个颗粒物的位置周长和面积之后，再显示在前端GUI上。后台算法流程如下（部分细节操作在后续步骤详解介绍）：



# 计数算法实现步骤

## 2.1图像去噪

对图像使用3x3矩阵中值滤波，即遍历图像中每一个像素点，每一个像素点的值替换为自身加上其八邻域内的中值，自己编写代码如下：

for i=1:m

for j=1:n

for ind=1:3

imax=min(i+1,m);

jmax=min(j+1,n);

imin=max(i-1,1);

jmin=max(j-1,1); J(i,j,ind)=median([I(i,jmax,ind),I(imax,j,ind),I(imax,jmax,ind),I(imax,jmin,ind),I(imin,jmax,ind),I(i,jmin,ind),I(imin,j,ind),I(imin,jmin,ind)]);

end

end

end

## 2.2大津法（最大类间方差法）二值化

大津法即最大类间方差法，简称OSTU的思路为：按照图像的灰度特性，先根据一个阈值可将图像分成背景和前景两部分，背景和前景之间的类间方差越大，说明构成图像的两部分的差别越大，而当部分前景错分类为背景或者部分背景错分类为目标会使得两部分类间方差变小。因此，类方差最大的分割意味着将前景和背景进行错分类的概率越小，即满足遍历所有阈值寻找一个使得类间方差最大。

算法步骤如下：

1. 遍历所有阈值，对于每一个阈值进行如下计算
2. 记N0为图中小于当前阈值的像素点的个数，N1为图中大于当前阈值的像素点的个数，M,N分别为图像的高和宽。则W0记为前景点占图像比例，同理W1记为背景点占图像比例，记μ0为前景点的平均灰度值，μ1为背景点的平均灰度值，

G记为类间方差，由此可得：

W0 = N0 ÷ （M x N）

W1 = N1 ÷ （M x N）

G = W0 \* （μ0 — μ）2 + W1 \* （μ1 — μ）2

1. 记录下循环遍历中使得G最大的阈值

自行编写计算OSTU阈值程序代码如下：

function [T0] = ostu(J1)

[m,n]=size(J1);

g0=0;

T0=0;

Jmin=min(J1(:));

Jmax=max(J1(:));

for T=Jmin:1/255:Jmax

w0=sum(J1(:)<=T)/(m\*n);%前景点比例

w1=sum(J1(:)>T)/(m\*n); %背景点比例

u0=mean(J1(J1<=T)); %前景点平均灰度值

u1=mean(J1(J1>T)); %背景点平均灰度值

g=w0\*w1\*(u0-u1)^2; %类间方差，用了一种等价形式

if g>g0 %记录下使得类间方差最大的阈值

g0=g;

T0=T;

end

end

end

## 开运算

## 孔洞填充（开运算前）

使用数学形态学的知识，此处的孔洞指二值图像内部八连通点阵形成组成的闭合圈内的像素点集，然后确定一个种子点，以该种子点为起点，使用四连通结构元对其进行膨胀运算，后续详细介绍膨胀运算，Matlab当中有实现该运算的函数：imfill(I,’hole’)

## 2.3.2 开运算：腐蚀运算

腐蚀运算的运算思路为，保证结构元真包含于图形的情况下对结构元进行运动，将其结构元的中心轨迹即点的集合记为腐蚀后的图像。

## 2.3.3开运算：膨胀运算

膨胀运算的运算思路为，保证结构元与图形有交集的情况下对结构元进行运动，将其结构元中心运动轨迹即点的集合记为膨胀后的图像。

Matlab对开运算，可使用imopen函数。

## 分水岭分割

## 距离变换

一幅图像经过二值化后可以分为前景和背景，距离变换即对图像中前景的像素点赋予首先对离它最近的背景像素点与其距离（Manhattan距离or欧氏距离），获得一个距离矩阵，即为新的图形，Matlab使用函数bwdist可以对图像求距离变换。

## 去除局部最小值

如果对距离变换后的图像直接进行分水岭算法，会有很多不必要得分割，为了去除这些分割，先使用imextendedmin将会只在我们希望分割的区块中间产生小点。然后我们再使用imimposemin函数进行滤波去除掉局部最小值。

## 2.4.3分水岭算法

分水岭算法（watershed）是一种借鉴了形态学原理的分割算法，在该方法中，将一幅图像看成一个拓扑地形图，其中灰度值f(x,y)对应地形高度值。高灰度值对应着山峰，低灰度值对应着山谷。水总是朝地势低的地方流动，指导某一局部低洼地才停下来，这个低洼处被称为吸水盆地，最终所有的水分份聚在不同的吸水盆地，吸水盆地之间的山脊被称为分水岭。水从分水岭流下时，它朝不同的吸水盆地流去的可能性是相等的。将这种想法应用于图像分割，就是要在灰度图像中找到不同的吸水盆地和分水岭，由这些不同的吸水盆地和分水岭组成的区域即为要分割的目标。

分水岭阈值选择算法可以看成一种自适应的多阈值分割算法，在图像梯度图上进行阈值选择时，经常遇到的问题是如何恰当的选择阈值。阈值若选的太高，则许多边缘会丢失或边缘出现破碎现象；阈值若选的太低，则容易产生虚假边缘，而且边缘变厚导致定位不精确。分水岭阈值选择算法可以避免这个缺点。

MATLAB图像处理工具箱中的watershed函数可用于实现分水岭算法，该函数的调用语法为：

L=watershed(f)

其中，f为输入图像，L为输出的标记矩阵，其元素为整数值，第一个吸水盆地被标记为1，第二个吸水盆地被标记为2，以此类推，分水岭被标记为0。

## 计数并统计信息

## 抑制边界连通区域

根据实验要求，需要去除边界的颗粒物，采用8连通边界判断，即一个连通区域如果存在像素点，其八个连通区域的有处于图像边界位置的，则判断这个连通区域为边界连通区域，清除该连通域。Matlab可以调用函数imclearborder(I,n)，其中I为图像，n为采用几连通域判断。

## 统计连通区域个数

使用bwlabel函数，对上步处理后的图像进行4连通判断，函数返回连通域的个数以及被标记后的图像，每一个连通域的像素值全部设置成编号值，即第一个连通域为1，第二个为2，以此类推。函数调用如下：

[L,num] = bwlabel(bw5,4);

其中L为标记后的图像，num为连通域个数。

## 统计每个连通域的面积

创建一个矩阵cell\_Area，行为连通域的个数，列为2，第一列为连通域的标号，第二列为连通域的面积。对于上一步处理后的图像L，遍历所有像素点，每一个像素点对应的值即标号值，则对该标号值的面积加一，自主完成代码如下：

%%%%%%%%计算每个细胞的面积

cell\_Area=[1:num;zeros(1,num)]';

for i=1:m

for j=1:n

if L(i,j)>0

cell\_Area(L(i,j),2)=cell\_Area(L(i,j),2)+1;

end

end

end

## 统计每个连通域的周长

使用bwboundaries函数，可以对图像进行连通域划分，获得每一个连通域的边界信息，函数返回一个细胞元组，其中每一个细胞元存储一个连通域的边界信息，格式为一个二列的矩阵，每一行为边界的坐标。同样创建一个矩阵cell\_L，用于存储每个连通域的周长，行为连通域的个数，列为2，第一列为连通域的标号，第二列为连通域的周长，对于使用了bwboundaries函数的返回值，进行遍历，细胞元组当中每一个细胞元的行数则为每一个连通域的周长，自主完成代码如下：

%%%%%%%%对每个小细胞在图片中编号，并获得每个细胞的周长

cell\_L=[];

for i=1:num

ix=round(0.5\*(max(B{i}(:,1))+min(B{i}(:,1))));%获得连通域横向的中点

iy=round(0.5\*(max(B{i}(:,2))+min(B{i}(:,2))));%获得连通域纵向的中点

cell\_L=[cell\_L;L(ix,iy),size(B{i}(:,1),1)];

%Text=string(L(ix,iy));

%text(iy,ix,Text,'horiz','center','color','white','fontsize',15)；%以上两个语句为后续在图片上为每一个连通域做标记的语句

end

## 获得每一个连通域的图片

因最后需要显示每一个连通域的图片，所以当前需要获得每个连通域的图片，思路为：

通过已经获得的连通域的边界，在图片上切取一个长为该连通域最长的横向长度，宽为连通域最长的纵向长度的矩形；

由于按照该方法切除之后，该矩形内可能存在其他连通域的部分像素，解决该问题的思路为：

在进行连通域标号之后，每个连通域的像素值为该连通域的标号值，故可以在切割时进行逻辑判断，切割范围内存在不属于当前连通域的像素值的时候，将其置为0，用一个细胞元组cell进行存储每一个连通域的图像矩阵，自主实现代码如下，：

%%%%%%%%获得每个小细胞的图片

for i=1:num

for x=min(B{i}(:,1)):max(B{i}(:,1))

for y=min(B{i}(:,2)):max(B{i}(:,2))

cell{i}(x-min(B{i}(:,1))+1,y-min(B{i}(:,2))+1,:)=(L(x,y)==i)\*I(x,y,:);

end

end

end

# 实验心得

本次实验当中，个人认为关键点有如下几点：

* 去噪参数和方法
* 通道的选取
* 开运算时结构元的大小（暂未解决）
* 进行分水岭算法前的距离变换
* 去除局部最小值

**去噪参数和方法：**

在对图像进行预处理时，首先对于去噪的方法，可选的有均值滤波和中值滤波，均值滤波优点为算法简单，速度快，但是对于噪声，不能完全去除，只能减弱，而且更重要的第一点，中值滤波能够保护图像的边缘，边缘信息对于连通域分割当中尤其重要，故最终选择中值滤波。

选择中值滤波之后，第二个问题也接踵而至，滤波模板大小的选择，而且考虑到后续还会有开运算，同样也有模板大小的选择，但是由于考虑到去噪和开运算解决的不同问题，去噪只是去除一些噪声点，另一角度可以优化二值化后的图像，而开运算是为了消除一些图像间的连接，并且去掉一些小的细胞，故第一步去噪的模板大小不应太大，否则会影响到后续开运算，本次实验当中，自行编写了模板为3x3的中值滤波。

**通道的选取：**

每次实验有三张图片，包含了石料和细胞，三张图片拥有的相同点是，背景为深色，颗粒物部分为浅色，不同点是，细胞显示为蓝色，石料分别显示为灰色和黄色，对于细胞而言，可以直接选取蓝通道进行后续操作，而对于石料而言，灰色和黄色选择任意通道的效果实验前难以判断。

故对于本次实验，本人提出一种解决方案，不采用任意通道值，直接获取三个通道中最亮的部分，这样可以解决存在显示颜色的问题，不只局限于红绿蓝三个通道，对于其他颜色也可以提取出信息。自主编写代码如下：

for i=1:m

for j=1:n

J1(i,j)=max([J(i,j,1),J(i,j,2),J(i,j,3)]);

end

end

后续对J1矩阵进行操作，经试验，三张图片采用该方法效果良好。

**开运算结构元的大小：**

在后续对图像进行开运算，去除细胞粘连以及微小细胞时，结构元的大小也决定了后续图像的效果，如果结构元太小，去除效果会非常差，甚至原本不属于边界的连通域会因为粘连去除强度不够被判断为边界连通域，而在后续实验被去除，严重影响结果；如果结构元太大，会导致连通域的面积缩水，影响实验结果的精度。为解决该问题，对于三张图片不断进行尝试，但由于三张图片分辨率差别较大，无法得到一个确定的大小，只能根据不同情况下进行调整。

**进行分水岭前需先进行距离变换：**

倘若直接对二值化后的图像进行分水岭操作，效果很差，究其原因，个人认为首先二值化的图像只存在两个值0和1，那么“水流”就全部流向为0的黑色区域，而当这个图像中的颗粒物十分密集的时候，存在颗粒物用开运算不能完全分割时，在分水岭算法中，这属于同一个水池，而且因为深度全部一致，也不能将这一个“水池”给划分开来。

解决方案便是上面步骤提到的距离变换，通过距离变换之后，每一个连通域图像的越靠近中间，其“地势”越低，那么如果有两个连通域存在部分粘连时，这些“水池”却不会有交集，这样就可以根据这些“水池”换划分连通域了。

**去除局部最小值：**

如果直接对距离变换后的图像进行分水岭操作，还存在着许多不必要的分割，而这些分割的存在便是局部最小值，反应在图像当中即是经过距离变换之后的图像当中有一些极小的“水池”，倘若不去掉这些最小值，整个分割也会将多出许多不必要的分割，严重影响统计信息的精度。

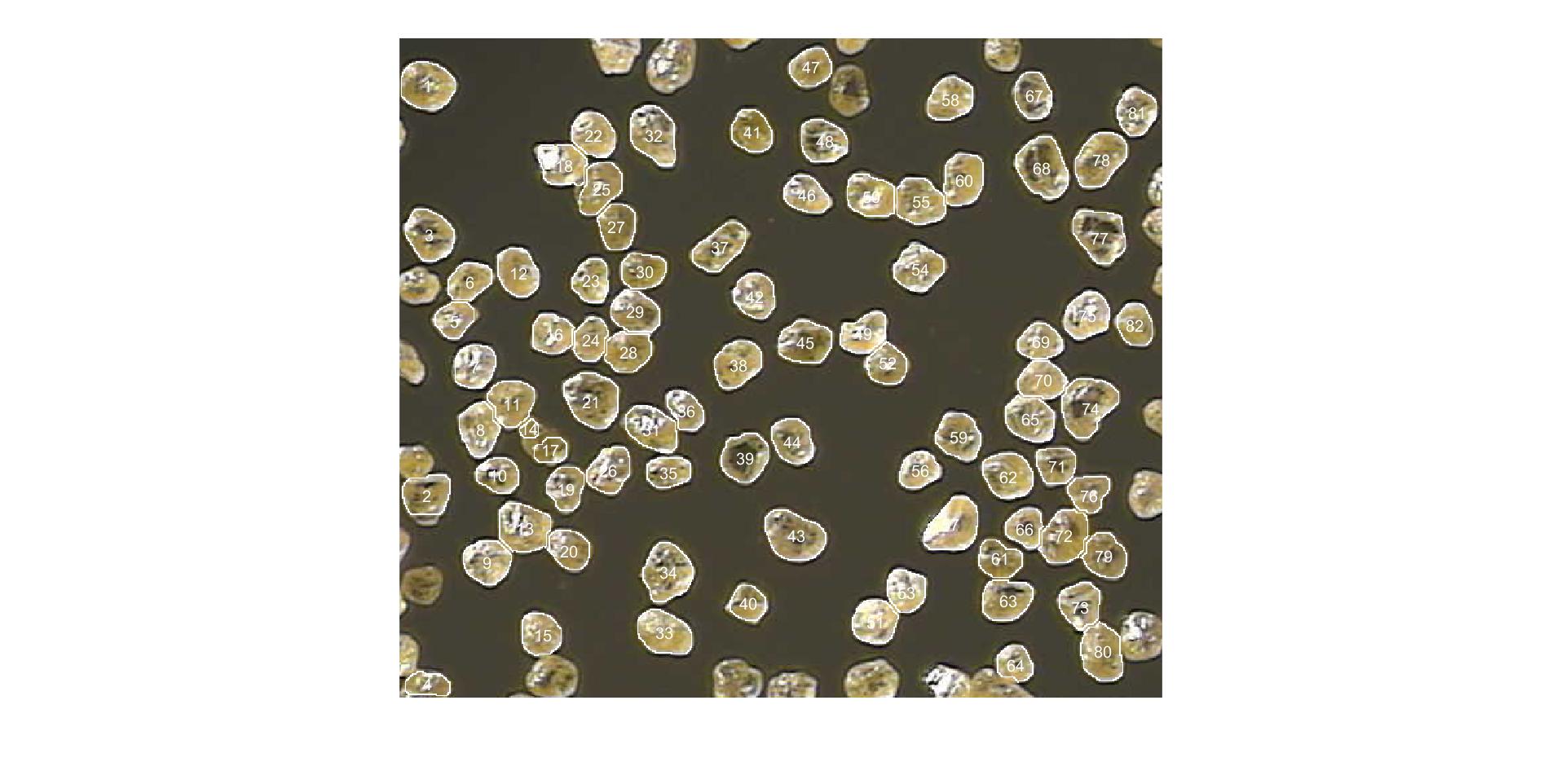
故解决方案即使用imextendedmin函数，标记出我们只想要得最小值，该函数使用格式为Mask = imextendedmin(D,n)，其中D为进行距离变换之后的图，n为最小值的上界，即在图片中标记出小于等于n的区域，获得的标记图像即为Mask。

之后再使用imimposemin函数，使其只会在我们标记的部分产生最小值，其他部分滤掉最小值，使用格式为D2 = imimposemin(D,Mask),Mask为标记出最小区域的图，D为进行距离变换后的图，D2为滤掉除最小区域外最小值后的图。

在经过以上两步之后，便没有了局部最小值的干扰，可以进行分水岭操作。

# 效果展示

* 分割结果展示



* GUI展示

