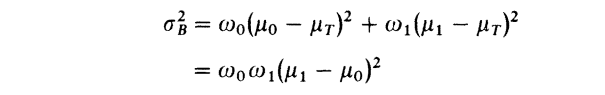
# 实现OTSU算法

该算法的核心思想是，由一个灰度阈值k来将0~255灰度划分成，0-k与k+1~255两类，通过灰度阈值来将图像二值化。

主要的优化函数为：



该函数值代表的意义是类间差异，即背景与目标物体的灰度差异。优化使该函数值最大的情况下所得到的k值，就是能够分割目标物体与背景的灰度阈值。

clc;

clear;

close all force;

img = imread('MRA.pgm');

figure();

imshow(img);

[l,h]=size(img);

%各个灰度值像素个数，xout是中心点的刻度

[n,xout]=hist(img(:),0:255);

count = n' ;

p = count / (l\*h); %归一化数据，每个灰度level的概率

max\_sig = 0;

c\_k = 0;

for k =1:256

% 统计类出现概率 这里0为背景，1为目标物体

w0 = sum(p(1:k));

w1 = sum(p(k+1:256));

% 统计类平均灰度

av0 = 0;

av1 = 0;

for i=1:k

av0 = av0 + i\*p(i);

end

u0 = av0 /w0;

for i =k+1:256

av1= av1 + i\*p(i);

end

u1 = av1/w1;

sig = w0\*w1\*(u1-u0)^2; %计算类间方差

if sig>max\_sig

max\_sig = sig; %使sig最大

c\_k = k; %记录此时的阈值

end

end

T = graythresh(img);

c\_k = (c\_k-1)/255;

% 根据所得阈值二值化

res1 = imbinarize(img,c\_k);

res2 = imbinarize(img,T);

figure();

subplot(131);

imshow(res1);

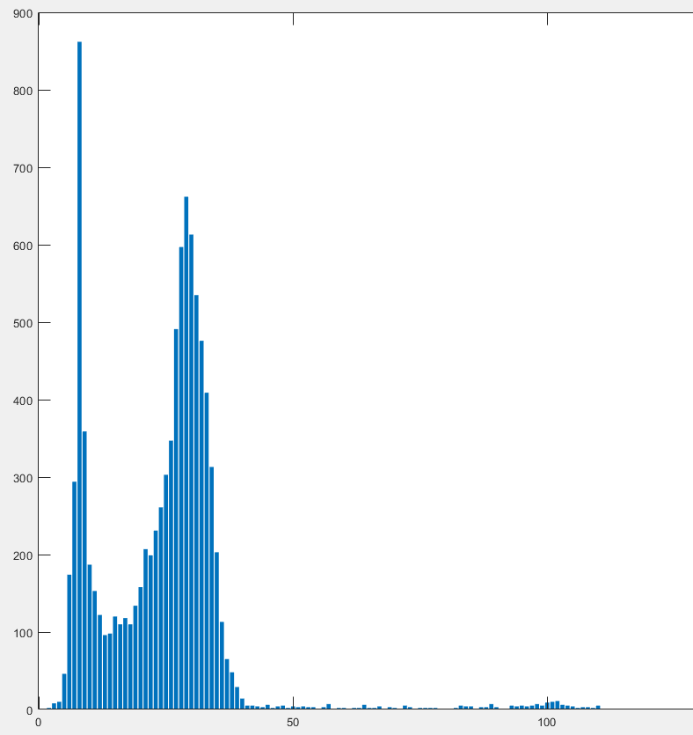
title('手动实现 ');

subplot(132);

imshow(res2);

title(' 官方api ');

原始图像的灰度分布图：

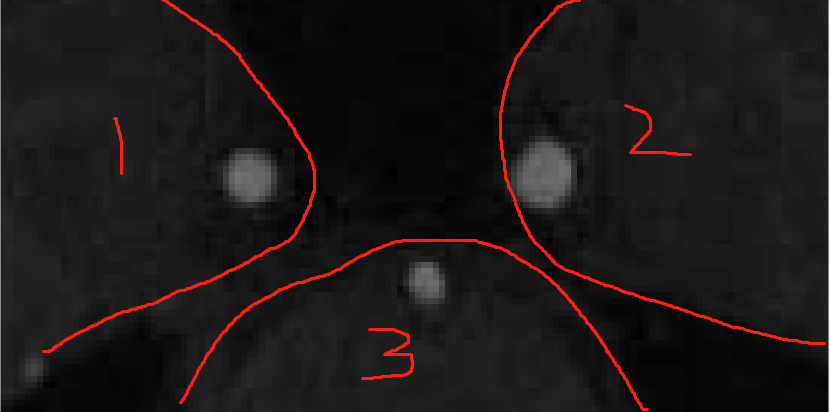


分割结果：



能够看到分割结果并不好，猜测主要的原因是：原图中的使得灰度类间差异最大的阈值k并不是我们想要的阈值点。如下图1,2,3区域的灰度大致一致，其他灰度区域大致一致，因此使得类间灰度差异最大的点只能将较大的区域灰度分割开来。而如果把三个白色的亮点与其他区域分割开，所使用的阈值k可能不能满足上面类间差异最大化的要求。

因此我们需要设置阈值的范围，在阈值范围内使得类内差异最小，从而得到我们想要的结果。由此引出RCOtsu算法。



# 实现RCOtsu算法

RCOtsu算法通过将灰度阈值rk限制在之内，在值域内遍历，使得类内差异

最小，所得rk即为所求的阈值。

在这里使用视觉判断来近似地估计参数。

阈值范围计算方法：





其中，H(i)为灰度为i时的累加频率，Hlb和Hhb通过面积估计来得到。

具体代码如下：

clc;

clear;

close all force;

img = imread('MRA.pgm');

figure();

imshow(img);

[l,h]=size(img);

%各个灰度值像素个数，xout是中心点的刻度

[n,xout]=hist(img(:),0:255);

count = n' ;

% bar(count);

p = count / (l\*h); %归一化数据，每个灰度level的概率

max\_sig = 0;

c\_k = 0;

all\_grey\_level = H( length(count) ,count);

%视觉估计函数 阈值范围

Hlb = 0.90\*all\_grey\_level;

Hhb = 0.95\*all\_grey\_level;

% rk的值域

r\_low = 0;

r\_high = 0;

flag = 0;

% 根据 Hlb 和Hhb 计算 r\_low 与 r\_high

for k=1:256

x=H(k,count);

if x>=Hlb &&flag ~=1

r\_low = k;

flag = 1;

end

if x>=Hhb && flag ~=0

r\_high = k ;

flag = 0;

break;

end

end

cost\_func = 0;

sig=[];

for k = r\_low:r\_high

% 统计类出现概率 变量尾数0为背景，1为目标物体

w0 = sum(p(1:k));

w1 = sum(p(k+1:256));

% 统计类平均灰度

av0 = 0;

av1 = 0;

for i=1:k

av0 = av0 + i\*p(i);

end

u0 = av0 /w0;

for i =k+1:256

av1= av1 + i\*p(i);

end

u1 = av1/w1;

%计算类内差异

temp=0;

for i=1:k

temp = temp+p(i)\*(i-u0)^2;

end

sig0 =temp/w0;

temp=0;

for i=k+1:256

temp = temp+p(i)\*(i-u1)^2;

end

sig1 =temp/w1;

temp = w0\*sig0+w1\*sig1;

sig = [sig,temp] ;

end

T = r\_low+find(sig==max(sig)) - 1 ;

T = T/255;

res = imbinarize(img,T);

subplot(121);

imshow(res);

subplot(122);

imshow(img);

% 累计频率计算函数

function [sum] = H(level,count)

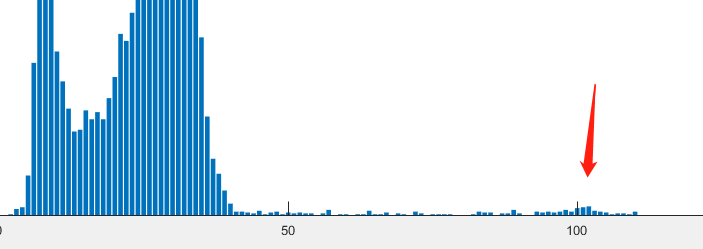
sum=0;

for i=1:1:level

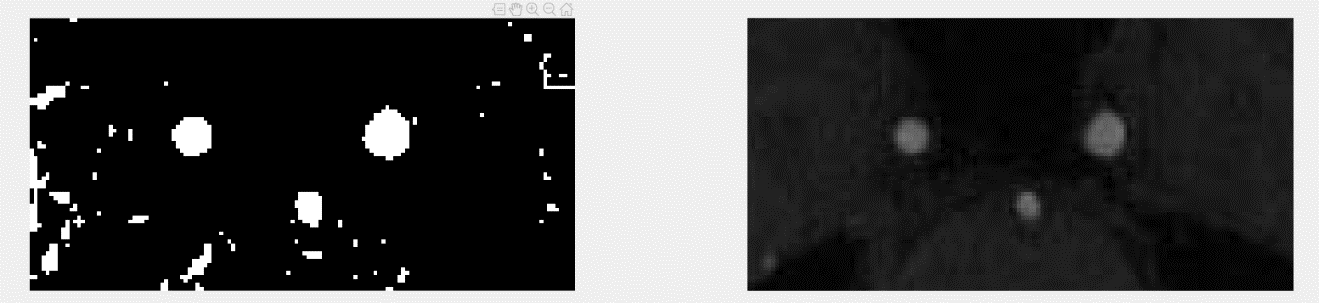
sum = sum +count(i);

end

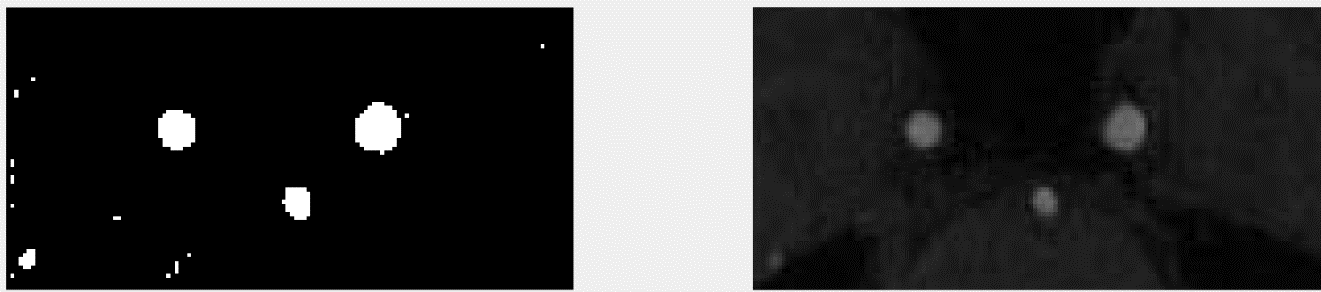
end

通过图像工具可以得到高亮点的灰度值大概在100左右，而根据灰度分布直方图能够看到100左右的灰度频数仅仅占很小一部分，因此初始时把背景所占范围设置成0.90~0.95

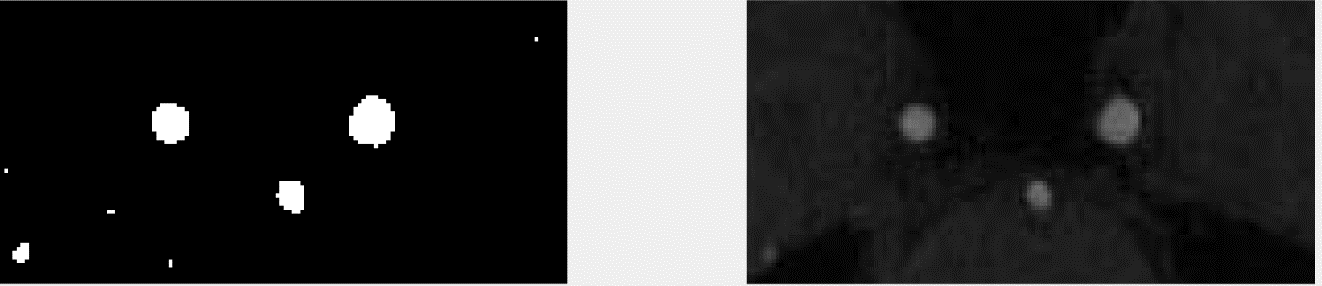
能够看到分割效果比Otsu算法好了很多，但仍有部分噪点，猜想可能是因为阈值范围设置略宽，导致一些相似灰度的像素也被划分为目标物体，因此继续调整阈值范围



当范围设置为0.965~0.985时，分割结果变好了，白色的噪点消失了很多，继续缩小阈值范围

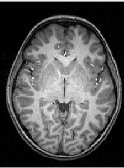
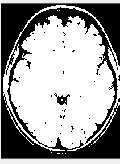


范围设置为0.972~0.985，可以看到白色噪点减少了，已经接近了我们想要的分割效果。



如果需要得到更加完美的分割结果，需要对血管面积进行更精确的估计来得到背景所占的范围，从而缩小rk的范围。

# 讨论

使用Otsu做头骨剖离，能够看到效果较好，但在血管分割的情况下效果就变差了，因此Otsu算法的分割效果会受到目标图像的灰度分布影响。

而RCOtsu算法能够通过设定目标灰度范围来限定灰度阈值到我们想要的灰度范围，能够得到较好的分割结果。而分割效果的好坏，在本次实验中用到的方法来看，主要依赖于对于背景所占图像比例的计算与估计精度，精度变高，就能把需要的灰度阈值逼近至我们想要的值。