

中南大学科研实习报告

数字图像处理



班 级: 生医1101班

姓 名: 文勋喆

学 号: 0145110818

时 间: 2014年 6月



一、 引言

数字图像处理(Digital Image Processing) 用计算机对图像信息进行处理的一门技术。数字图像处理利用计算机对图像进行各种处理的技术和方法。

早期的图像处理的目的是改善图像的质量，它以人为对象，以改善人的视觉效果为目的。图像处理中，输入的是质量低的图像，输出的是改善质量后的图像，常用的图像处理方法有图像增强、复原、编码、压缩等。

20世纪20年代，图象处理首次得到应用，60年代末，图象处理技术不断完善，逐渐成为一个新兴的学科。利用数字图象处理主要是为了修改图形，改善图象质量，或是从图像中提起有效信息，还有利用数字图象处理可以对图像进行体积压缩，便于传输和保存。数字图象处理主要研究以下内容：傅立叶变换、小波变换等各种图象变换；对图像进行编码和压缩；采用各种方法对图像进行复原和增强；对图像进行分割、描述和识别等。随着技术的发展，数字图象处理主要应用于通讯技术、宇宙探索遥感技术和生物工程等领域。

二、 目的

1. 用 MATLAB 或其他的语言来实现数字图像处理方面的一些操作；
2. 熟悉 MATLAB7.1 的一些基本函数及与数字图像处理相关的函数；
3. 熟悉图形用户界面 (GUI)，并用其来编写界面；
4. 熟悉数字图像处理课程中的一些知识点，如图像灰度变化，直方图，图像增强，滤波，图像复原，形态学处理，图像边缘检测，图像放大和缩小等等，并能用 MATLAB 实现以上的功能；
5. 掌握从简单到复杂的方法，一步一步的实现功能，并能耐心排错，养成合作互助精神。



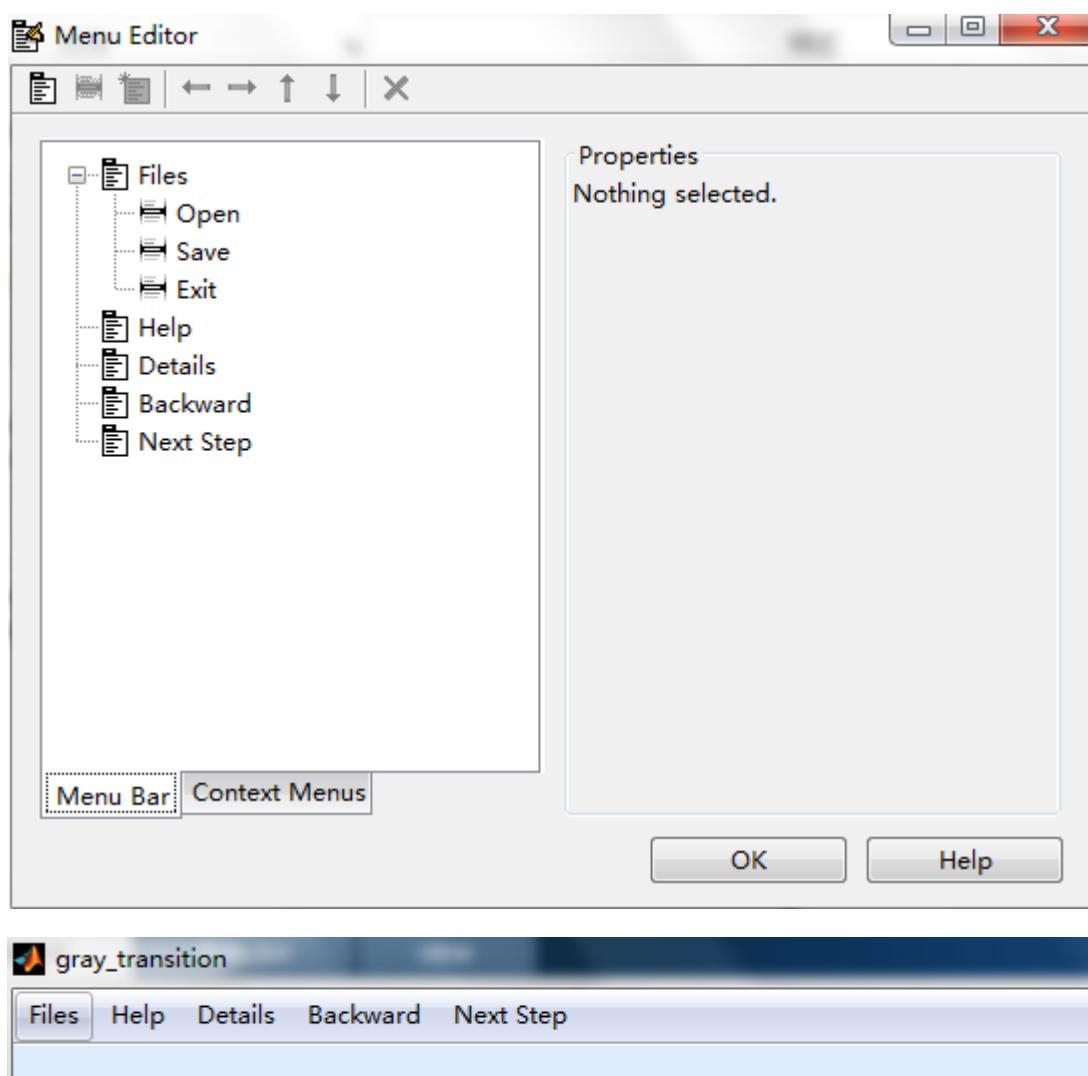
三、任务

1. 实现图像的打开，显示，保存操作，实现图像的旋转；
2. 实现图像的灰度变换（图像反转，对数反转，幂次反转）；
3. 实现图像的大小变换；
4. 实现图像的直方图处理及直方图均衡化；
5. 实现图像的空域和频域增强，实现均值滤波，维纳滤波，中值滤波及巴特沃斯低通、高通滤波；
6. 实现图像的加噪，如椒盐噪声、高斯噪声等；
7. 实现图像的形态学运算，如开运算，闭运算，膨胀运算，腐蚀运算等操作；
8. 实现图像的边缘检测，如 Robert 算子边缘检测，Sobel 算子边缘检测，Prewitt 算子边缘检测，Log 算子边缘检测，Canny 算子边缘检测；
9. 实现整个界面的设计，编排，最终达到一种比较完美的效果。

四、内容

(一) 菜单栏设计 [main_menu]

通过 Menu Editor 创建菜单，如图一所示，主要有“文件”、“编辑”、“帮助”、“关于”等选项，运行如图二所示，其中它们又各有自己的子菜单。以下分别介绍各个菜单的实现及部分源代码。



1、图像的读取和保存

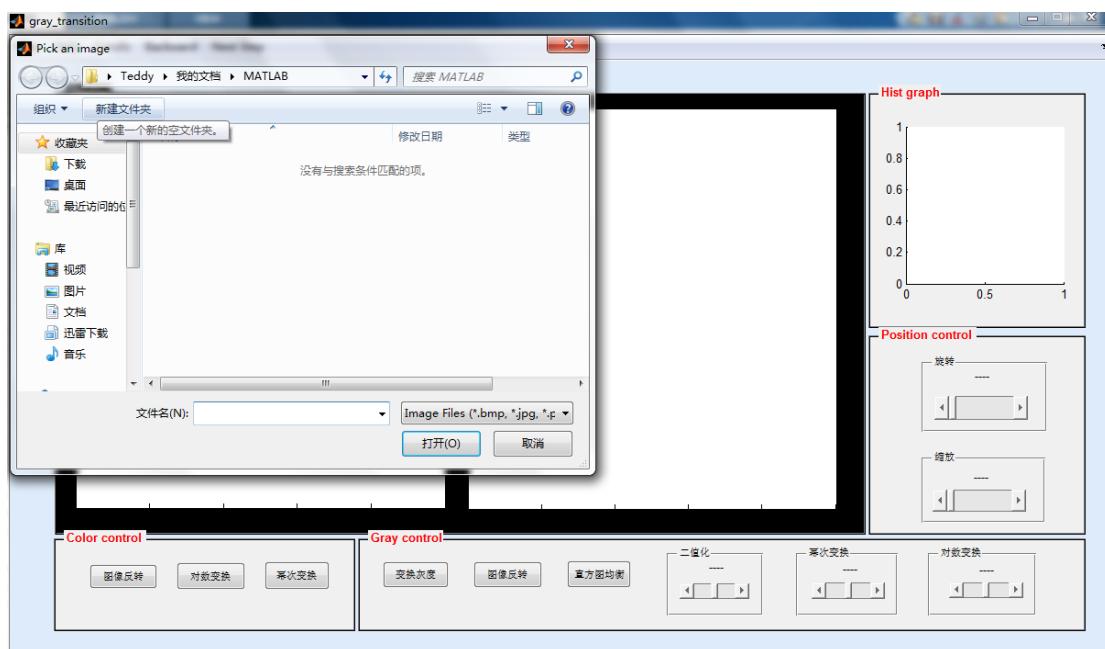
(1) 利用“文件”菜单中的“打开”实现图片的读取：

利用matlab中“uigetfile”、“imread”“imshow”实现图像文件的读取与显示，实现程序如下：

```
[filename, pathname] = uigetfile( ...
{'.bmp;*.jpg;*.png;*.jpeg', 'Image Files (*.bmp, *.jpg,
*.png,*.jpeg)'}, ...
{'.*', 'All Files (*.*)'}, ...
'Pick an image');
if isequal(filename,0) || isequal(pathname,0),
return;
end
axes(handles.axes1);
fpath=[pathname filename];
img_1=imread(fpath);
```



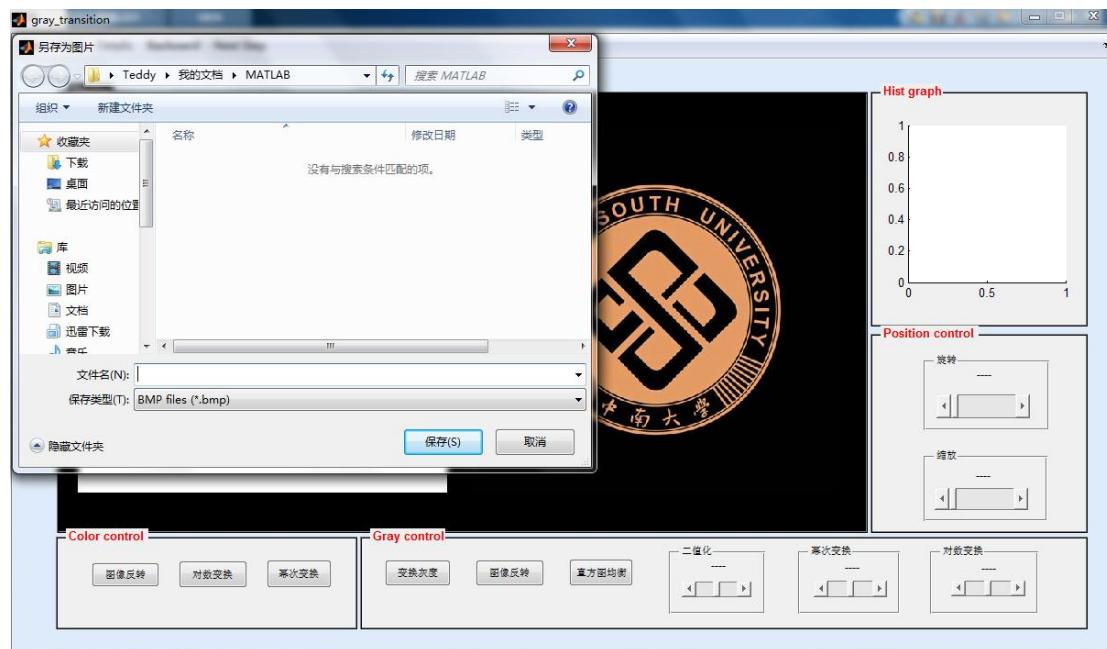
```
imshow(img_1);
```



(2) 图像保存

利用“uiputfile”、“imwrite”函数实现图像文件的保存，实现程序如下：

```
[filename, pathname] = uiputfile({'*.bmp', 'BMP files'; '*.png', 'PNG files'}, '另存为图片');
if isequal(filename, 0) || isequal(pathname, 0)
return;
else
fpath=fullfile(pathname, filename);
end
imwrite(f, fpath);
```

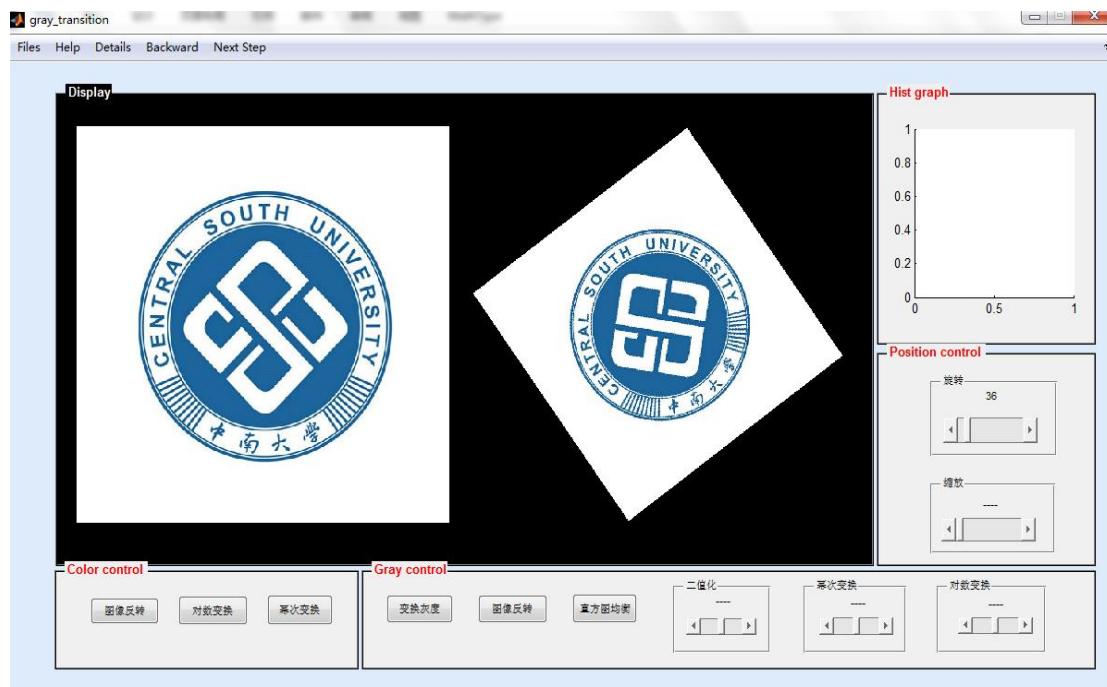


2、图像的编辑

(1) 图像旋转

利用slider控件，对旋转角度控制，并使用“imrotate”函数实现图像文件的旋转，实现程序如下：

```
b=get(handles.slider2,'value');  
f=imrotate(img_1,b);  
axes(handles.axes2);  
imshow(f);  
set(handles.text4,'String',b);
```

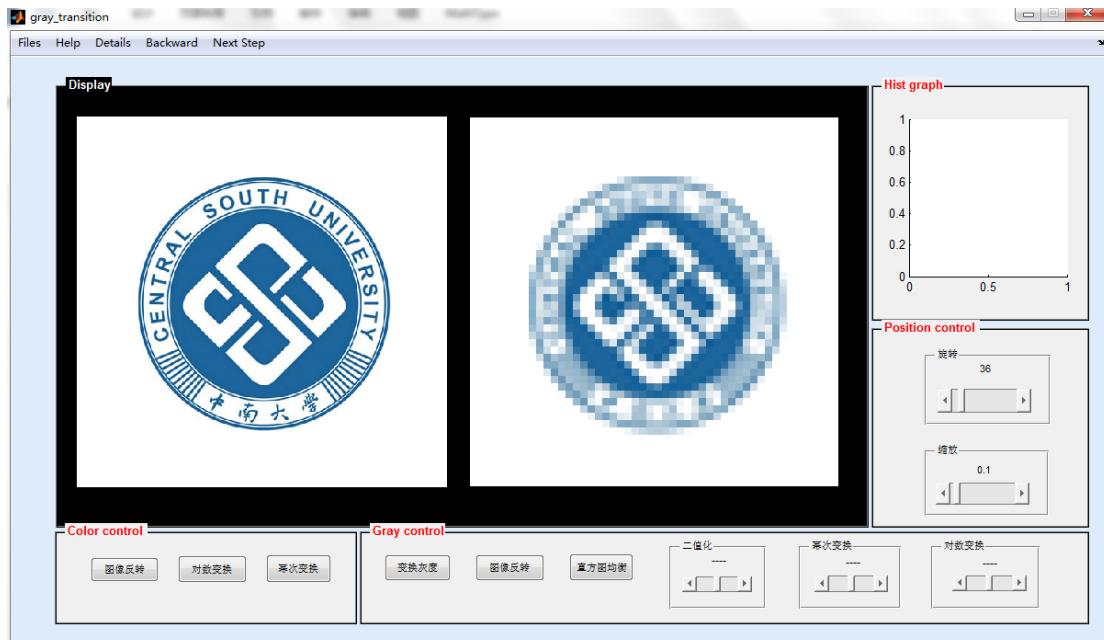




(2) 图像缩放

利用slider控件，对缩放倍率控制，并使用“imresize”函数实现图像文件的旋转，实现程序如下：

```
b=get(handles.slider5,'value');  
f=imresize(img_1,b);  
axes(handles.axes2);  
imshow(f);  
set(handles.text5,'String',b);
```



(2) 灰度变换 [gray_transition]

由于在 matlab 中较多的图像处理函数支持对灰度图像进行处理，故对图像进行灰度转化十分必要。在灰度变换中又可以有不同的处理方法，比如图像反转，对数变换，幂次变换，二值化，直方图均衡等，其中每种处理方法都将有自己的特色。

1. 图像反转:

将图像各像素实现反转，实现程序如下：

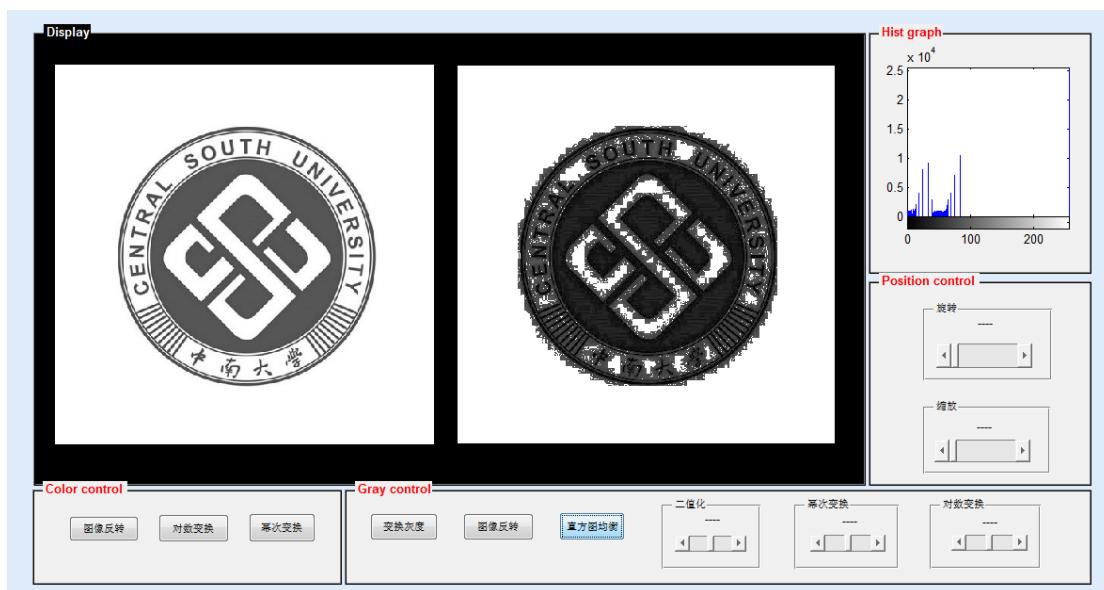
```
g=rgb2gray(img_1);  
gray=im2double(g);  
f=1-gray;  
axes(handles.axes2);imshow(f)
```



2. 直方图均衡:

直方图均衡是使原直方图变换为具有均匀密度分布的直方图，然后按该直方图调整原图像的一种图像处理技术。直方图均衡的实验结果为：

```
gray=rgb2gray(img_1);
gray_e=histeq(gray,256);
axes(handles.axes2);imshow(gray_e);
axes(handles.axes3);imhist(gray_e);
```



3. 二值化:

将图像作阈值二值化分割，实现程序如下：

```
gray_a=rgb2gray(img_1);
```



```
gray=im2double(gray_a);
b=get(handles.slider6,'value');
th=b;
gray_thd=im2bw(gray,th);
axes(handles.axes2);imshow(gray_thd);
axes(handles.axes3);imhist(gray_thd);
set(handles.text1,'String',b);
```



4. 幂次变换:

将图像作幂次变化，实现程序代码和演示结果如下：

```
g=rgb2gray(img_1);
gray=double(g);
b=get(handles.slider10,'value');
f=gray.^b;
axes(handles.axes2);imshow(uint8(f));
axes(handles.axes3);imhist(uint8(f));
set(handles.text2,'String',b);
```





5. 对数变换:

对数校正是输入图像像元亮度值(灰度)按对数函数关系变换为输出图像。对数变换实现了图像灰度扩展和压缩的功能。它扩展低灰度值而压缩高灰度值, 让图像的灰度分布更加符合人的视觉特征。实现程序代码和演示结果如下:

```
g=rgb2gray(img_1);
gray=double(g);
b=get(handles.slider11,'value');
f=b*log(1+gray);
axes(handles.axes2);imshow(uint8(f));
axes(handles.axes3);imhist(uint8(f));
set(handles.text3,'String',b);
```



(三) 噪声滤波 [noise_filter]

1. 图像加噪

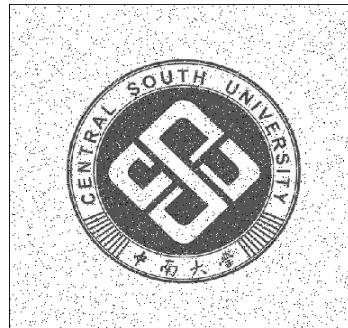
图像的加噪处理利用 `imnoise` 函数对原图像进行, “gaussian”、“salt & pepper”、“poisson”, “speckle” 分别对应不同的噪声类型。本试验中, 各噪声函数的方差等参数都为器默认值。并利用 `radiobutton` 对控件实行分组集中管理, 其实现代码与效果展示为:



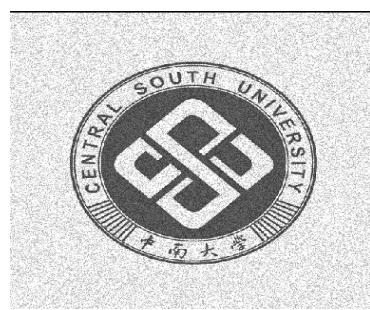
(a) 高斯噪声



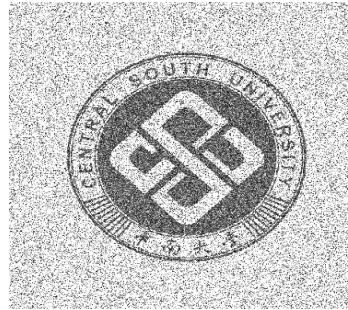
(b) 椒盐噪声



(c) 乘性噪声



(d) 瑞利噪声



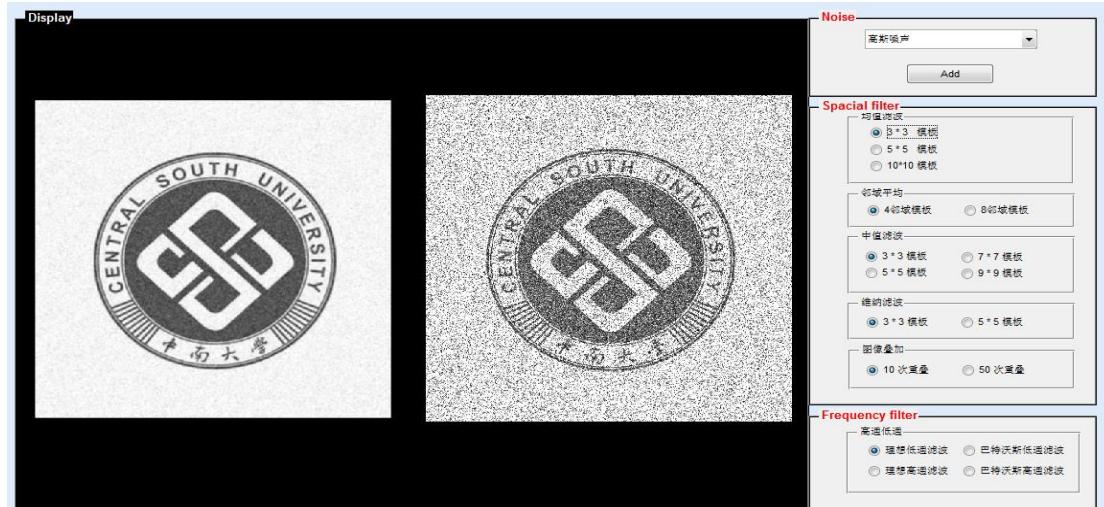
2. 图像滤波

图像利用 `filter2` 函数对原图像的噪声图进行滤波处理，不同的滤波器类型的滤波原理不同，因此滤波效果也有所不同。对每一种滤波器选择的模板大小不同，在滤波的效果也与所不同。

1. 均值滤波:

均值滤波也称为线性滤波，其采用的主要方法为邻域平均法。线性滤波的基本原理是用均值代替原图像中的各个像素值，即对待处理的当前像素点 (x, y) ，选择一个模板，该模板由其近邻的若干像素组成，求模板中所有像素的均值，再把该均值赋予当前像素点 (x, y) ，作为处理后图像在该点上的灰度的 $g(x, y)$ ，即 $g(x, y) = \frac{1}{m} * \sum f(x, y)$ ， m 为该模板中包含当前像素在内的像素总个数。其实现代码如下：

```
f=filter2(fspecial('average',3),noise_pic);
```





2. 邻域滤波:

邻域滤波构建一个关于中心像素的 4×4 邻域范围，在单元中进行滤波。其实现代码如下：

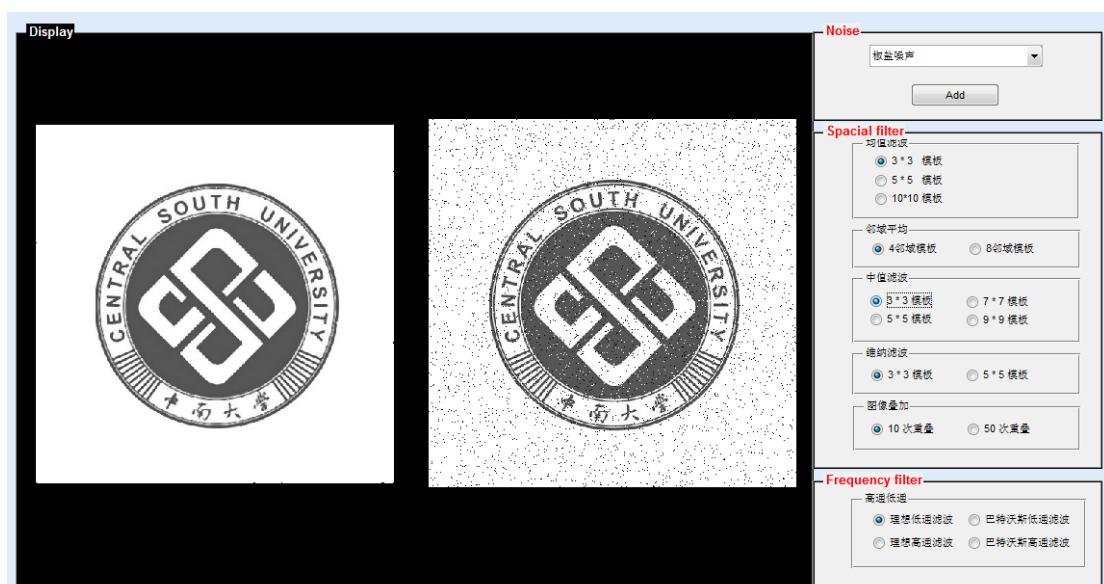
```
M4=[0 1 0;1 0 1;0 1 0];  
M4=M4/4;  
f=filter2(M4,noise_pic);  
axes(handles.axes1); imshow(f);
```



3. 中值滤波:

中值滤波原理是在一个单元化的邻域范围之内，利用像素值中间的一位数的值代替中心位置的像素单元，达到滤波的目的，其适合对椒盐噪声的滤除。其实现代码如下：

```
f=medfilt2(noise_pic);
```





4. 维纳滤波:

维纳是用来解决从噪声中提取信号的一种过滤（或滤波）方法。这种线性滤波问题，可以看做是一种估计问题或一种线性估计问题。其实现代码如下：

```
f=wiener2(noise_pic,[3 3]);
```



5. 理想低通滤波器:

低通滤波器容许低频信号通过，但减弱(或减少)频率高于截止频率的信号的通过。对于不同滤波器而言，每个频率的信号的减弱程度不同。理想低通滤波器出现了比较明显的振铃效应，对原图像有干扰。其实现代码如下：

```
D=40;  
Fuv=fftshift(fft2(gray));  
[p,q]=size(Fuv);  
x0=floor(p/2);y0=floor(q/2);  
for u=1:p  
    for v=1:q  
        d=sqrt((u-x0)^2+(v-y0)^2);  
        if(d>=D)  
            h(u,v)=0;  
        else  
            h(u,v)=1;  
        end  
    end  
end  
Guv=h.*Fuv;  
f=ifftshift(Guv);
```



```
f=uint8(real(ifft2(f)));
axes(handles.axes1);imshow(f);
```



6. 巴特沃斯低通滤波器:

巴特沃斯滤波器的特点是通频带内的频率响应曲线最大限度平坦，没有起伏，而在阻频带则逐渐下降为零。在振幅的对数对角频率的波特图上，从某一边界角频率开始，振幅随着角频率的增加而逐步减少，趋向负无穷大。二阶巴特沃斯滤波器能有效的消除振铃效应。其实现代码如下：

```
D=40;N=2;
Fuv=fftshift(fft2(gray));
[p,q]=size(Fuv);
x0=floor(p/2);y0=floor(q/2);
for u=1:p
    for v=1:q
        d=sqrt((u-x0)^2+(v-y0)^2);
        if d==0
            h(u,v)=0;
        else
            h(u,v)=1/(1+(D/d)^(2*N));
        end
    end
end
Guv=h.*Fuv;
f=ifftshift(Guv);
f=uint8(real(ifft2(f)));
axes(handles.axes1);imshow(f);
```

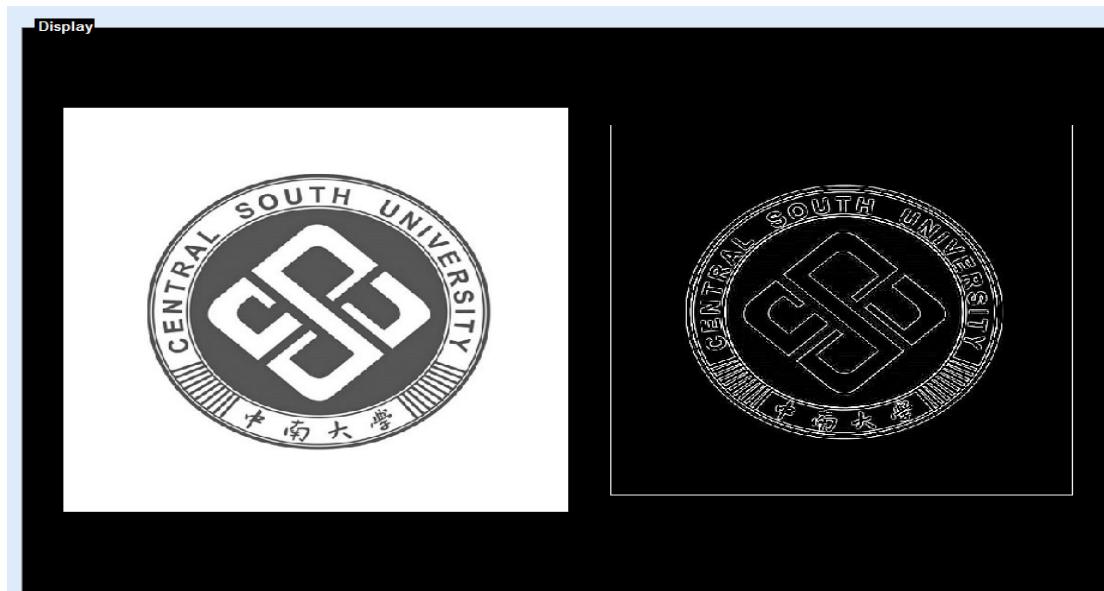


(四) 边缘检测 [edge_detection]

图像的边缘检测处理利用“edge”函数对原图像进行，不同的边缘检测模型。本试验中，分别采用了各检测方式的默认阈值。最后运用矩阵对图像进行检测，得到的效果有待评判。操作界面利用 radiobutton 对控件实行分组集中管理，其实现代码与效果展示为：

1. Linear

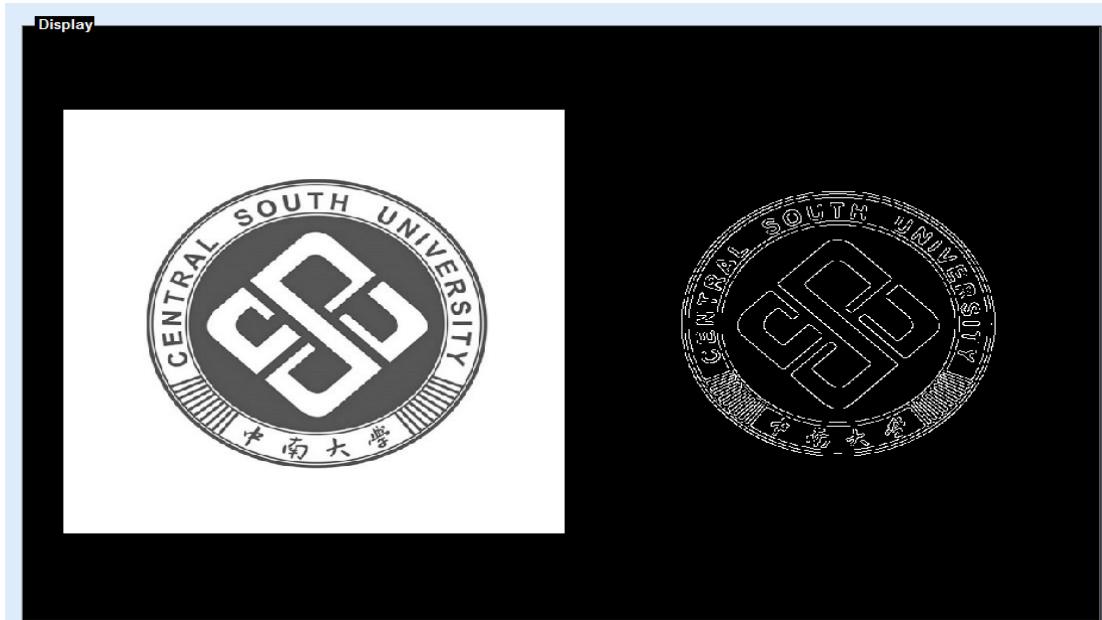
```
h=[ -1 -1 -1;  
     -1  8 -1;  
     -1 -1 -1;];  
f=imfilter(g,h);  
axes(handles.axes2);f=uint8(f);imshow(f);
```





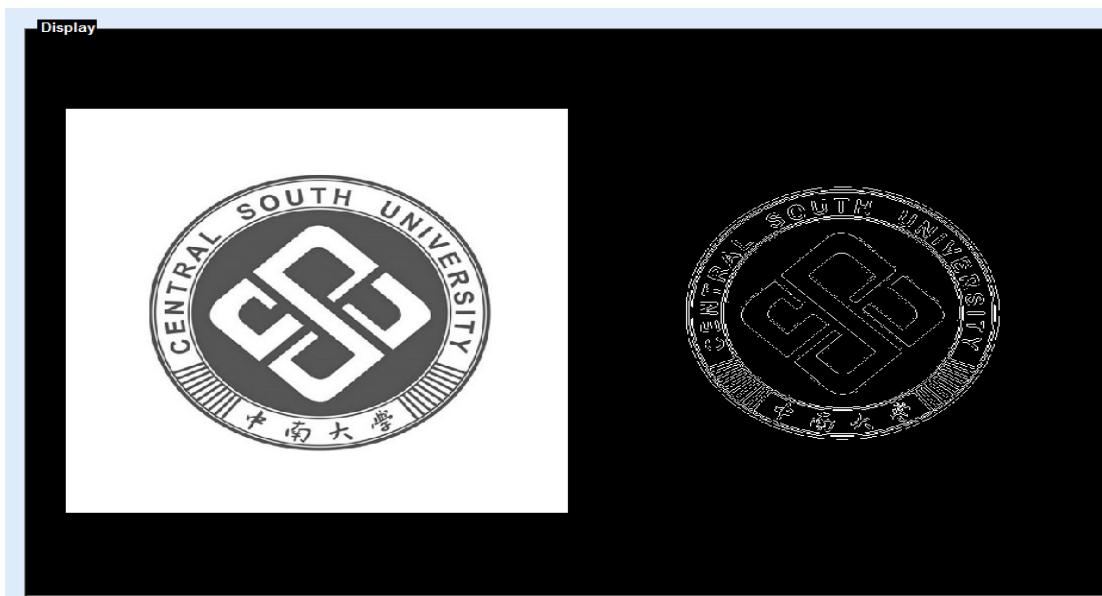
2. Canny

```
f=edge(g, 'canny', [0.03,0.06]);  
axes(handles.axes2);  
imshow(f);
```



3. Prewitt

```
f=edge(g, 'prewitt', 0.02);  
axes(handles.axes2);  
imshow(f);
```





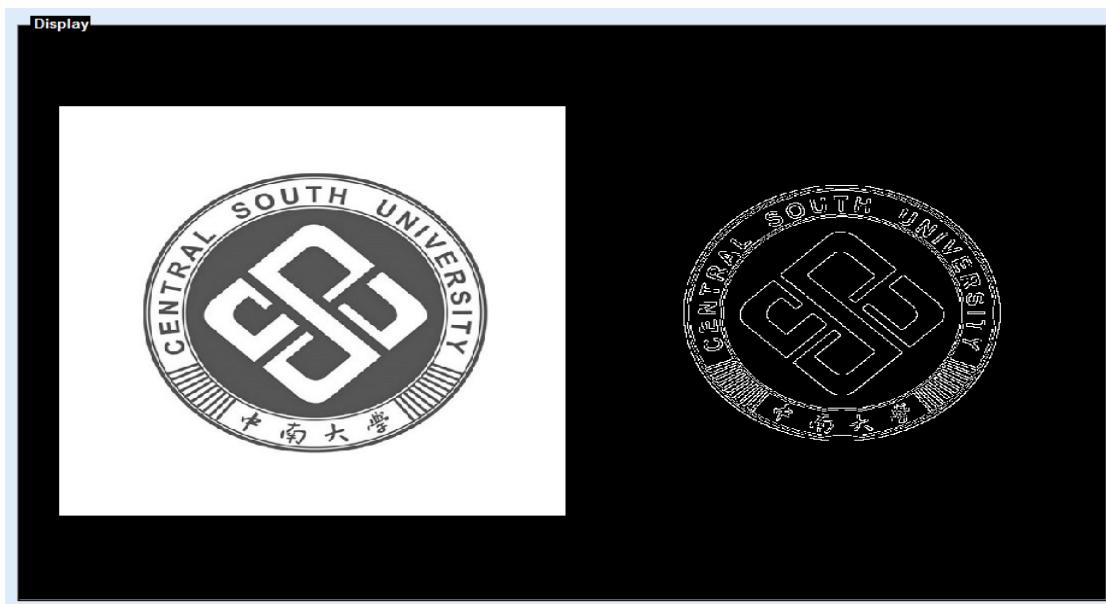
4. Log

```
f=edge(g, 'log', 0.002, 1.7);  
axes(handles.axes2);  
imshow(f);
```



5. Robert

```
f=edge(g, 'roberts', 0.07);  
axes(handles.axes2);  
imshow(f);
```





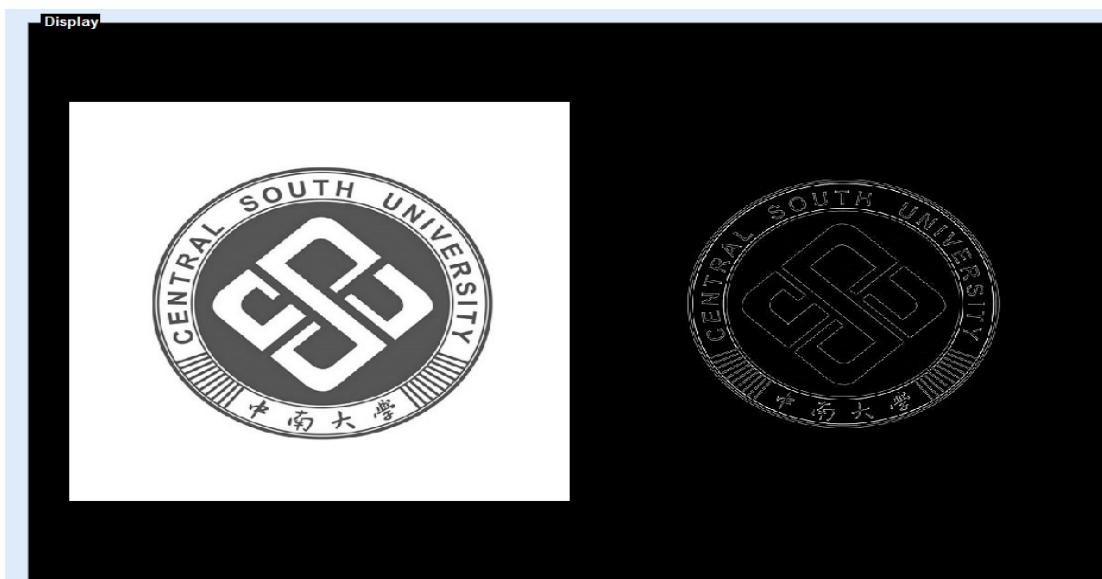
6. Sobel

```
f=edge(g, 'sobel', 0.07);  
axes(handles.axes2);  
imshow(f);
```



7. Laplacian 模板

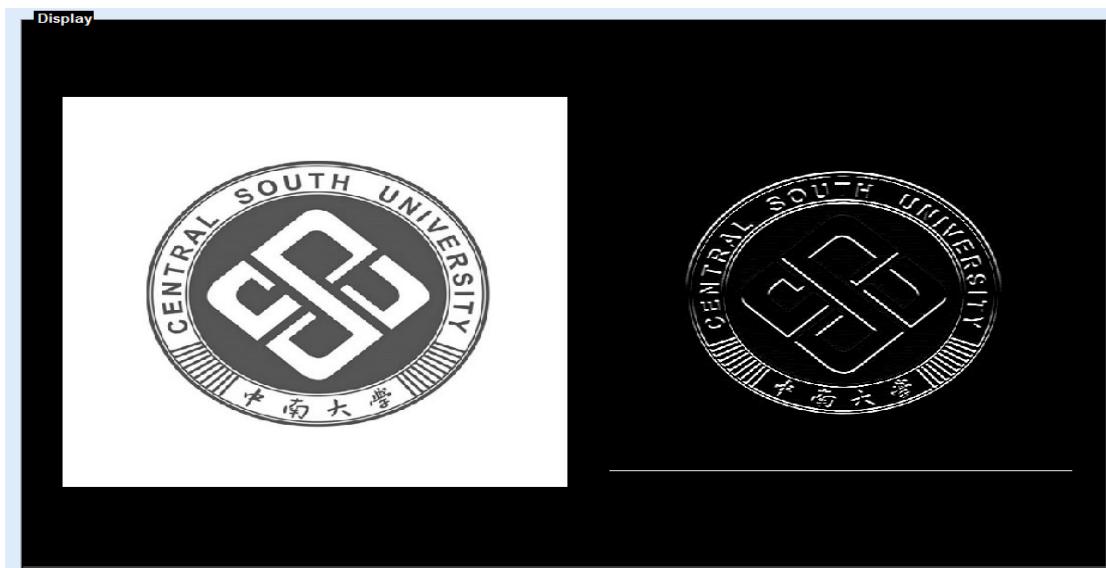
```
h=fspecial('laplacian');  
f=imfilter(g,h);  
axes(handles.axes2);  
imshow(f);
```



8. X-方向边缘检测

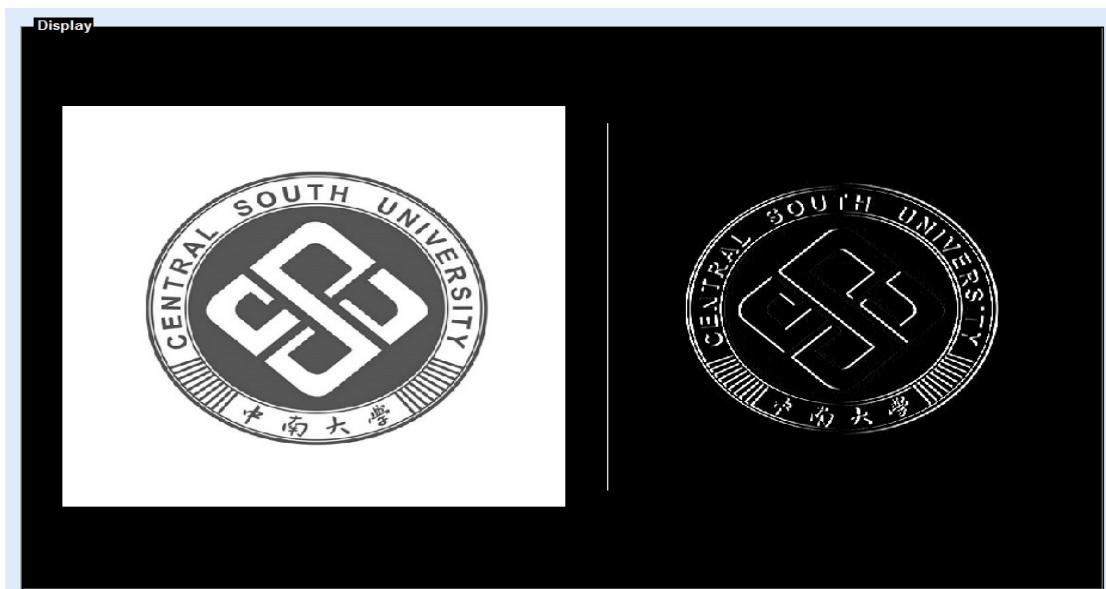


```
h1=[-1 0 1;
     -2 0 2;
     -1 0 1;];
f=imfilter(g,h1);
axes(handles.axes2);
imshow(f);
```



9. Y-方向边缘检测

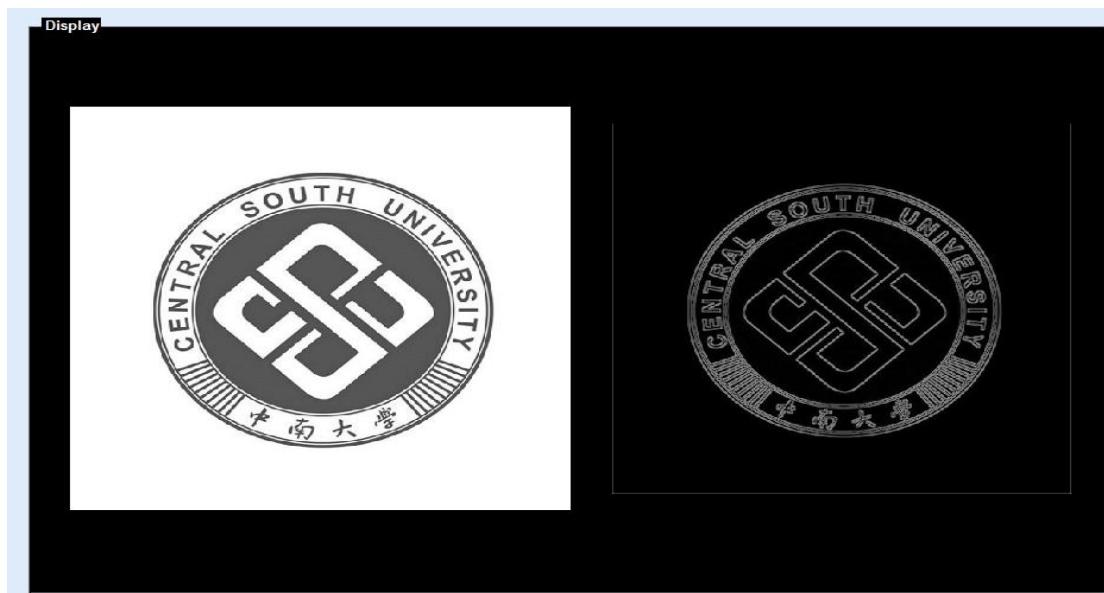
```
h2=[1 2 1;
     0 0 0;
     -1 -2 -1;];
f=imfilter(g,h2);
axes(handles.axes2);
imshow(f);
```





10. X-Y 方向边缘检测

```
h1=[1 2 1;
     0 0 0;
     -1 -2 -1;];
h2=[-1 0 1;
     -2 0 2;
     -1 0 1;];
h3=[-1 -2 -1;
     0 0 0;
     1 2 1;];
h4=[1 0 -1;
     2 0 -2;
     1 0 -1;];
k1=imfilter(g,h1);
k2=imfilter(g,h2);
k3=imfilter(g,h3);
k4=imfilter(g,h4);
f=double(k1)+double(k2)+double(k3)+double(k4);
m=uint8(f/4);
f=m;
axes(handles.axes2);
imshow(f);
```



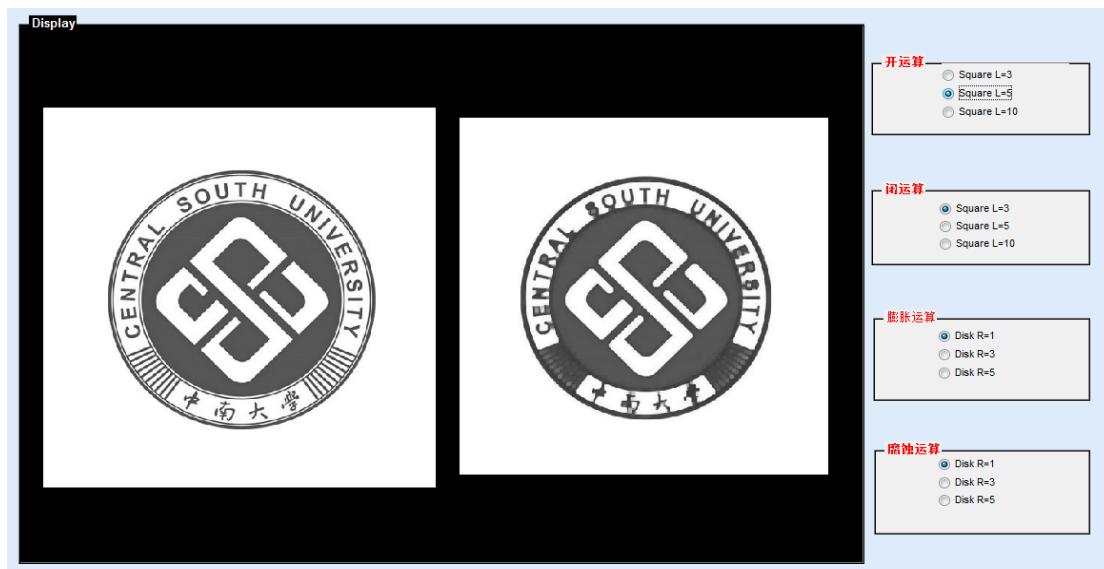


(五) 形态学 [morph_process]

形态学运算是一个很重要的处理工具，设计有：开运算、闭运算、腐蚀及膨胀。形态学的处理效果与所选择的结构元素的构建有关系。结构元素的构建包括其形状和大小，不同的结构元素对图像处理的结果也有所不同。

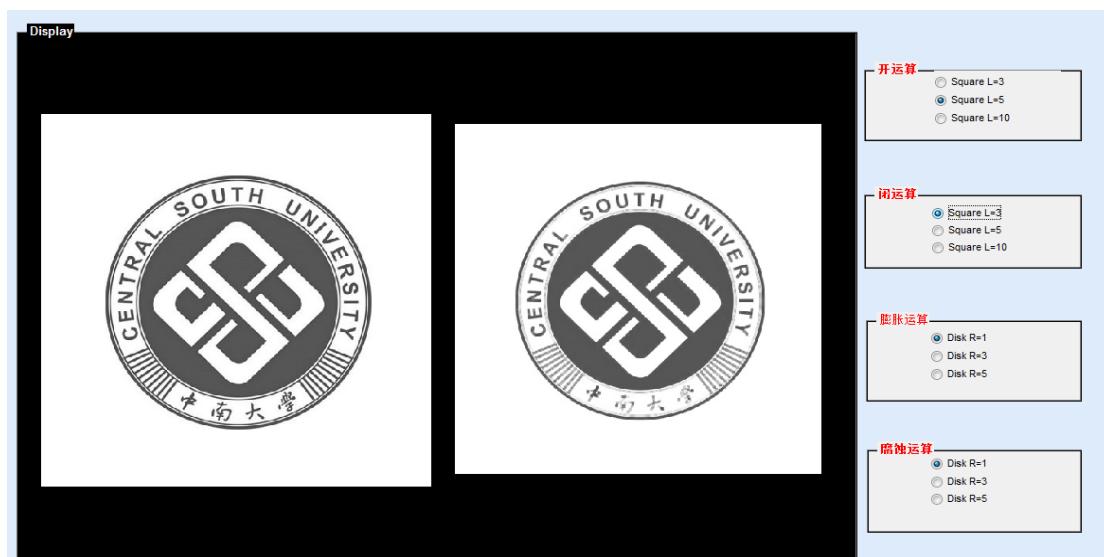
1. 开运算，其实现代码与结果展示如下：

```
st=strel('square',3);  
f=imopen(gray,st);
```



2. 闭运算，其实现代码与结果展示如下：

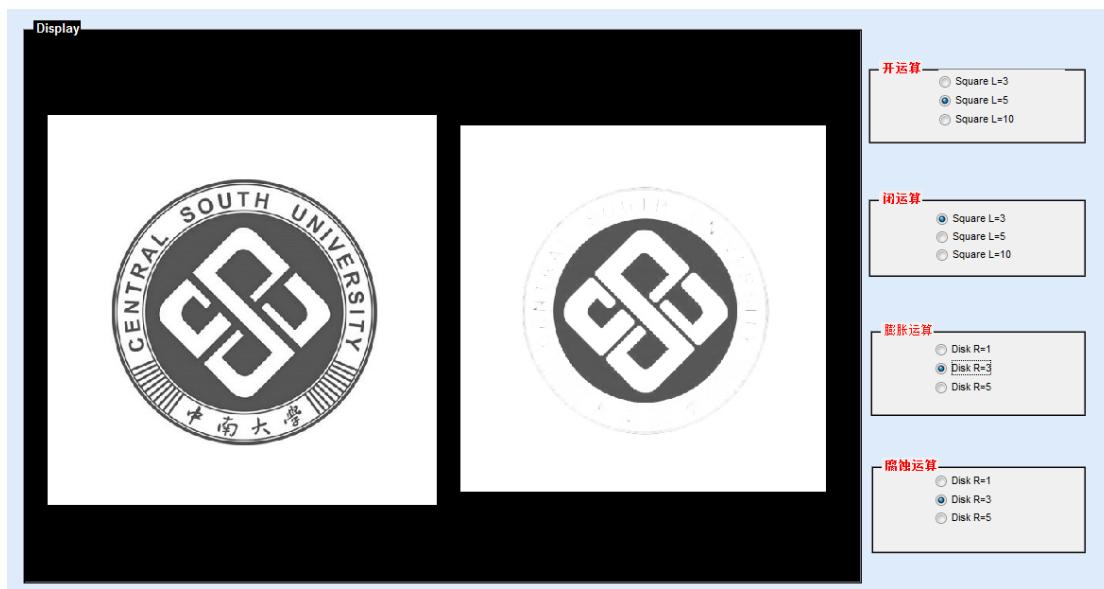
```
st=strel('square',3);  
f=imclose(gray,st);
```





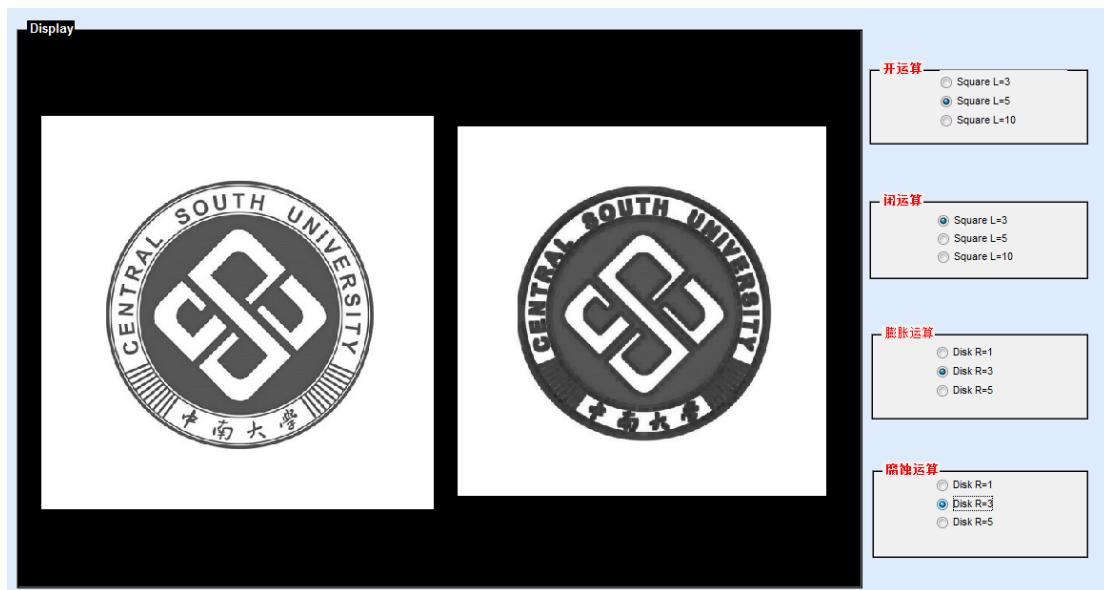
3. 膨胀运算，其实现代码与结果展示如下：

```
st=strel('disk',1);  
f=imdilate(g,st);
```



4. 腐蚀运算，其实现代码与结果展示如下：

```
st=strel('disk',1);  
f=imerode(g,st);
```





五、附加功能

在满足了实习所要求的基本任务后，我们在平常的学习中举一反三，勇于创新，利用平时所学，将整个工程设计得更加完备。
我们在原有基础上加入了图像分割和图像增强的内容。

1. 伪彩色处理

伪彩色处理就是把黑白图像的灰度值映射成相应的彩色。伪彩色处理分为灰色分层法和灰度变换法。

其实现代码与结果展示如下：

```
for i = 1:m
    for j = 1:n

        if Image(i,j) < Len/16
            R(i,j) = 0;
            G(i,j) = 0;
            B(i,j) = 0;

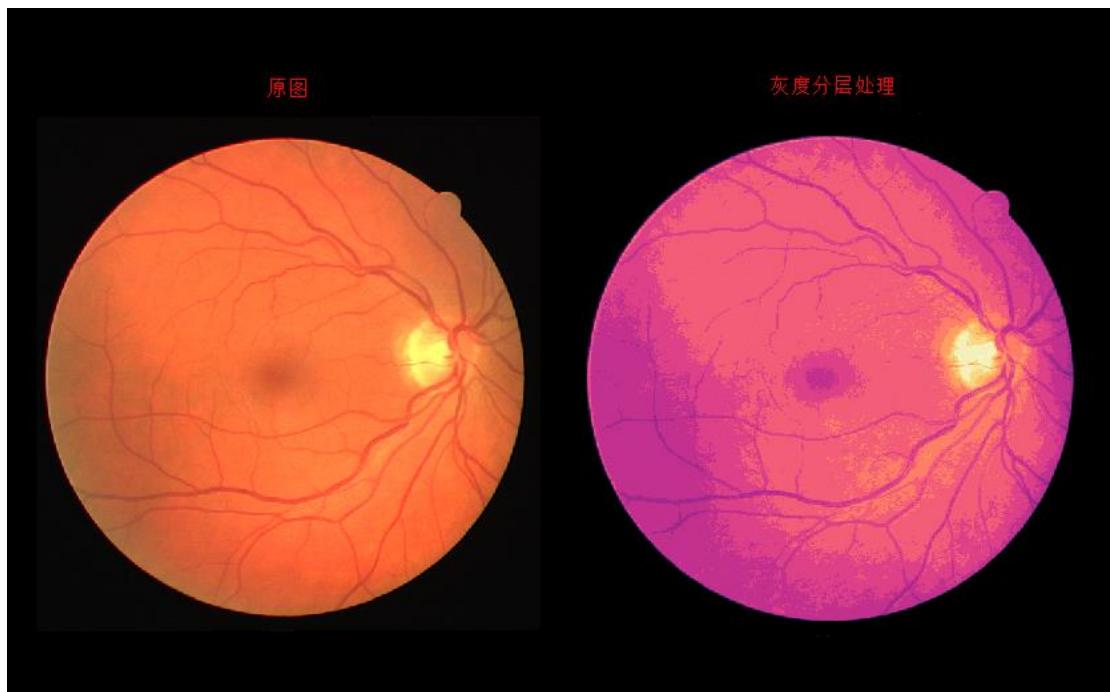
        elseif Image (i,j) < Len/8
            R(i,j) = 26;
            G(i,j) = 16;
            B(i,j) = 63;

        elseif Image (i,j) < 3*Len/16
            R(i,j) = 45;
            G(i,j) = 24;
            B(i,j) = 84;

            ...
        else
            R(i,j) = 255;
            G(i,j) = 255;
            B(i,j) = 255;

Re_image= repmat(I,[1 1 3]);

for i = 1:m
    for j = 1:n
        Re_image(i,j,1) = R(i,j);
        Re_image(i,j,2) = G(i,j);
        Re_image(i,j,3) = B(i,j);
    end
end
```



2. 多尺度顶帽变换

多尺度顶帽变换将图像的微小细节或图像的明暗特征抽取出来，以增加图像亮图案的特征，减弱暗图像的特征，初步增强的视网膜图的直方图中像素经截取调整后，分布基本满足正态分布，由此可以利用高斯函数来拟合，结合 3σ 原则作图像的增强。

```
g=rgb2gray(f);
Frw=zeros(128,128);
Frb=zeros(128,128);
Fdw=zeros(200,200);
Fdb=zeros(200,200);
maxWTH_i=0;
maxBTH_i=0;
maxDBTH_i=0
maxDWTH_i=0;

se=strel('disk',3);
j=[0 1 0;1 1 1;0 1 0];
B=padarray(j,[3,3]);

for i=1:13
Bi=imdilate(j,se,'full');
disp(Bi);
open=imopen(g,Bi);
close=imclose(g,Bi);
```



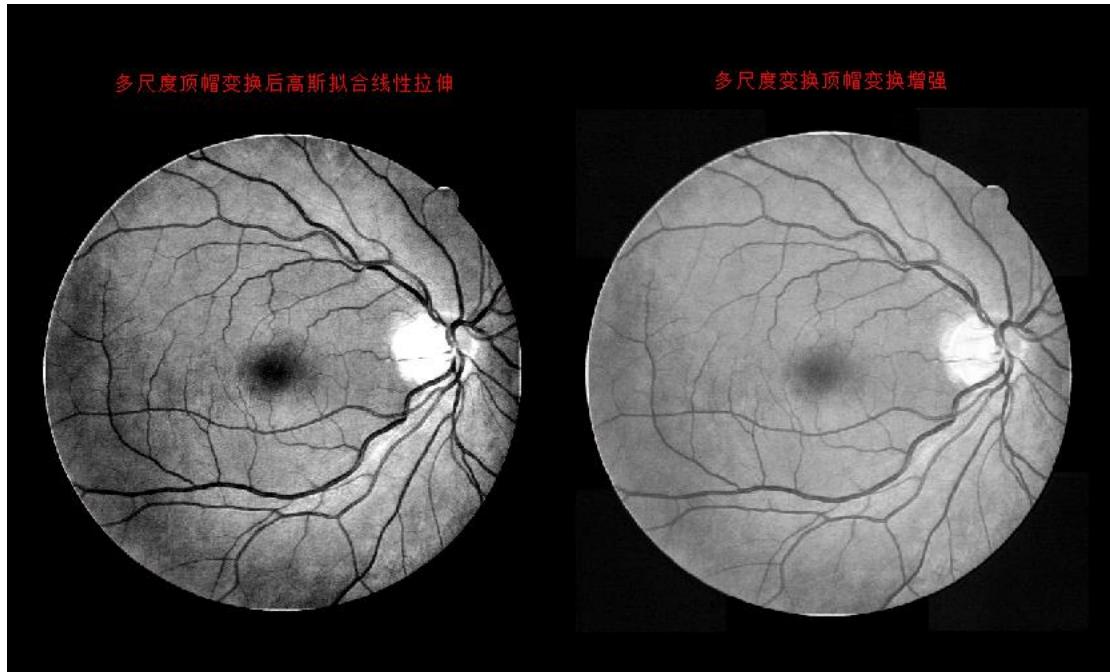
```
WTH_i=imsubtract(g,open);
BTH_i=imsubtract(close,g);
Bi_1=imopen(Bi,se);
open_1=imopen(g,Bi_1);
close_1=imclose(g,Bi_1);

WTH_i1=imsubtract(g,open_1);
BTH_i1=imsubtract(close_1,g);
DWTH_i=imsubtract(WTH_i1,WTH_i);
DBTH_i=imsubtract(BTH_i1,BTH_i);

T0=sum(WTH_i);
arg_1=sum(T0)./(numel(WTH_i));
T1=sum(BTH_i);
arg_2=sum(T1)./(numel(BTH_i));
T3=sum(DWTH_i);
arg_3=sum(T3)./(numel(DWTH_i));
T4=sum(DBTH_i);
arg_4=sum(T4)./(numel(DBTH_i));

if arg_1>maxWTH_i;
    maxWTH_i=arg_1;
    Frw=WTH_i;
end
if arg_2>maxBTH_i;
    maxBTH_i=arg_2;
    Frb=BTH_i;
end
if arg_3>maxDWTH_i;
    maxDWTH_i=arg_3;
    Fdw=DWTH_i;
end
if arg_4>maxDBTH_i;
    maxDBTH_i=arg_4;
    Fdb=DBTH_i;
end
end

Fen=g+Frw+Fdw-Frb-Fdb;
imshow(Fen)
```



3. 大津算法

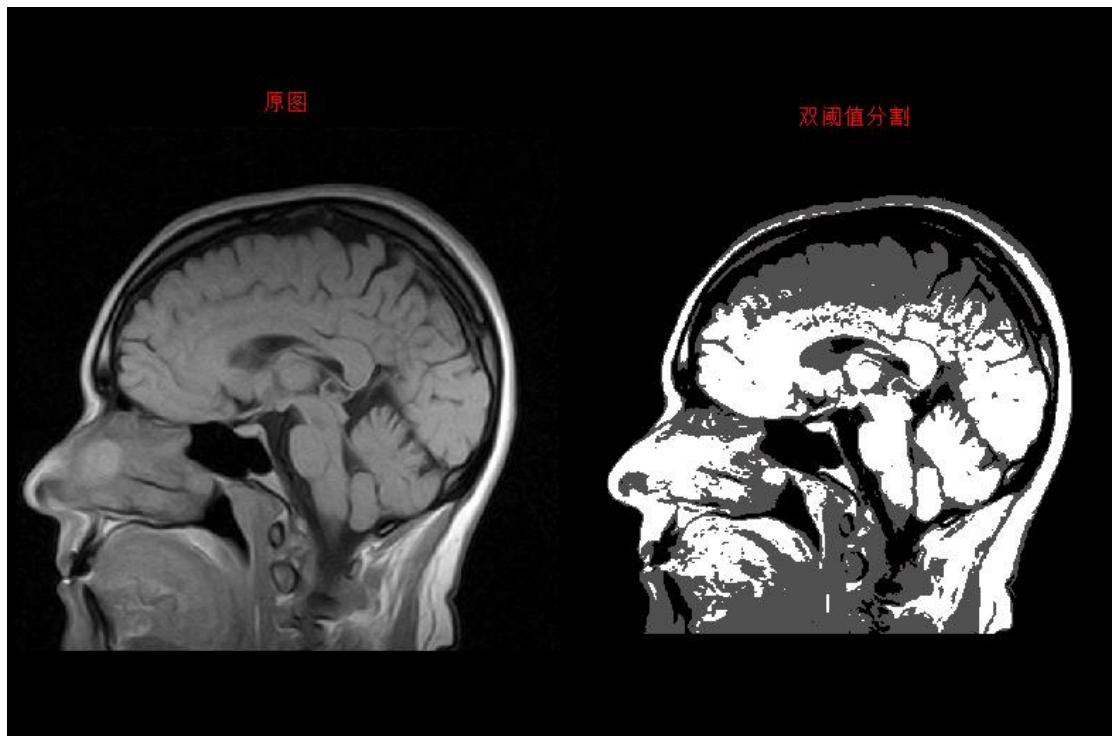
以前的分割方法存在计算量大, 计算复杂等问题, 在很多情况下, 直方图由高斯分布估计出来, 但高斯分布有时与实际模型联系不紧密。阈值将原图象分成前景, 背景两个图象。当取最佳阈值时, 背景应该与前景差别最大, 关键在于如何选择衡量差别的标准而在 otsu 算法中这个衡量差别的标准就是最大类间方差。设使用某一个阈值将灰度图像根据灰度大小, 分成目标部分和背景部分两类, 在这两类的类内方差最小和类间方差最大的时候, 得到的阈值是最优的二值化阈值。

```
imggray=im;
[m,n]=size(imgray);
N=m*n;
numgray=zeros(1,256);
P=zeros(1,256);
w0=zeros(256,256);
u0=zeros(256,256);
w1=zeros(256,256);
u1=zeros(256,256);
w2=zeros(256,256);
u2=zeros(256,256);
vardouble=zeros(256,256);
for i=1:m
    for j=1:n
        numgray(imgray(i,j)+1)=numgray(imgray(i,j)+1)+1;
    end
end
```



```
for i=0:255
    P(i+1)=numgray(i+1)/N;
end
meanlevel=0;
for i=0:255
    meanlevel=meanlevel+i*P(i+1);
end
for k1=0:255;
    for k2=(k1+1):255
        for kk=0:k1
            w0(k1+1,k2+1)=w0(k1+1,k2+1)+P(kk+1);
            u0(k1+1,k2+1)=u0(k1+1,k2+1)+kk*P(kk+1);
        end
        for kk=k1:k2
            w1(k1+1,k2+1)=w1(k1+1,k2+1)+P(kk+1);
            u1(k1+1,k2+1)=u1(k1+1,k2+1)+kk*P(kk+1);
        end
        for kk=k2:255
            w2(k1+1,k2+1)=w2(k1+1,k2+1)+P(kk+1);
            u2(k1+1,k2+1)=u2(k1+1,k2+1)+kk*P(kk+1);
        end

        vardouble(k1+1,k2+1)=w0(k1+1,k2+1)*(u0(k1+1,k2+1)-meanlev
el)^2+w1(k1+1,k2+1)*(u1(k1+1,k2+1)-meanlevel)^2+w2(k1+1,k
2+1)*(u2(k1+1,k2+1)-meanlevel)^2;
    end
end
maxvar=max(vardouble);
for i=1:256
    for j=i+1:256
        if vardouble(i,j)==max(maxvar)
            tk1=i-1;
            tk2=j-1;
            return
        end
    end
end
```

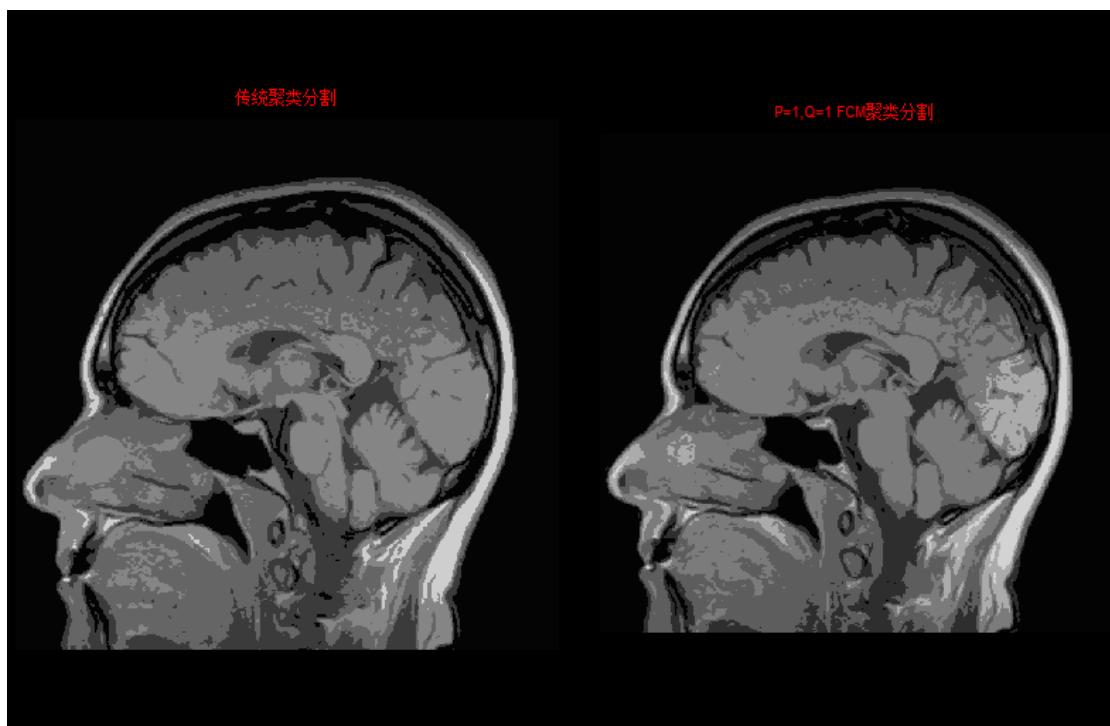


4. 聚类分割

聚类分割通过模糊 c-均值 (FCM) 聚类实现图像的分割。动态聚类方法的目的是把 n 个样本划分到 c 个类别中的一个，使各样本与其所在类均值的误差平方和最小。

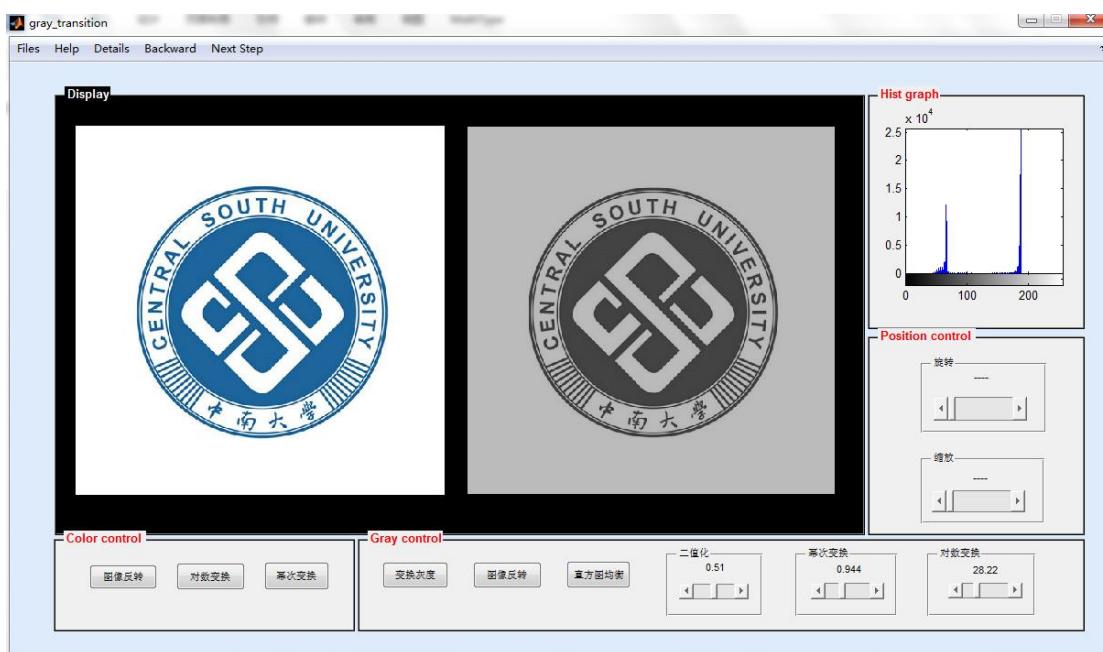
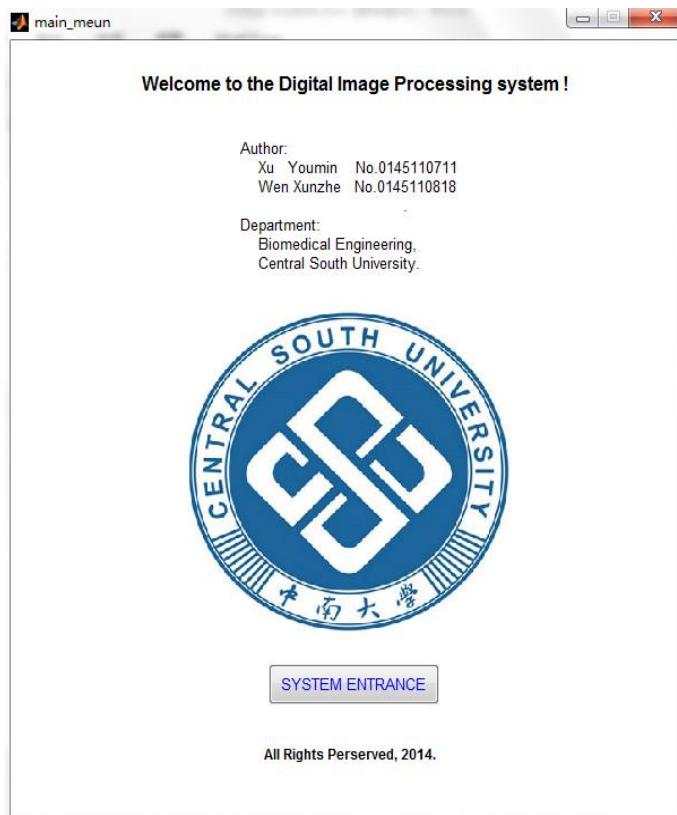
```
imgray=rgb2gray(im);
[m,n]=size(imgray);
c=5;
pm=2;
u=zeros(c,m*n);
umin=zeros(c,m*n);
x=zeros(1,m*n);
for i=1:m
    for j=1:n
        x(1,(i-1)*n+j)=imgray(i,j);
    end
end
v=[250.55 255.55 160.5 30.2 56.3];
for i=1:c
    sumd=zeros(1,m*n);
    for k=1:c
        sumd=sumd+((x-v(i))./(x-v(k))).^2;
    end
    u(i,:)=1./sumd;
end
minJ=+inf;
```

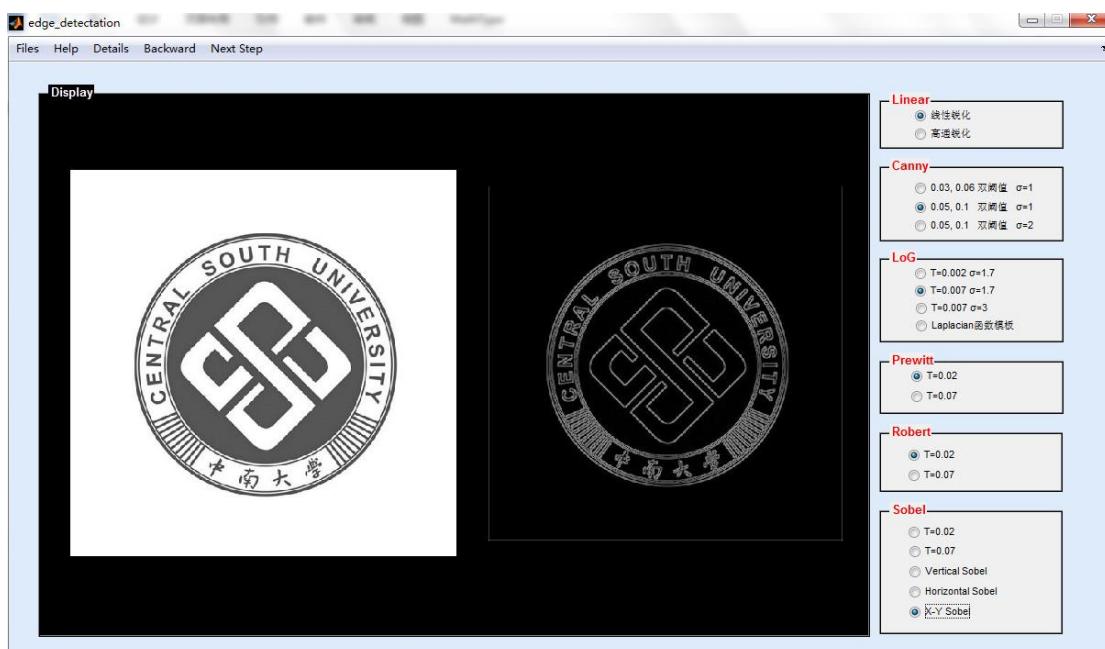
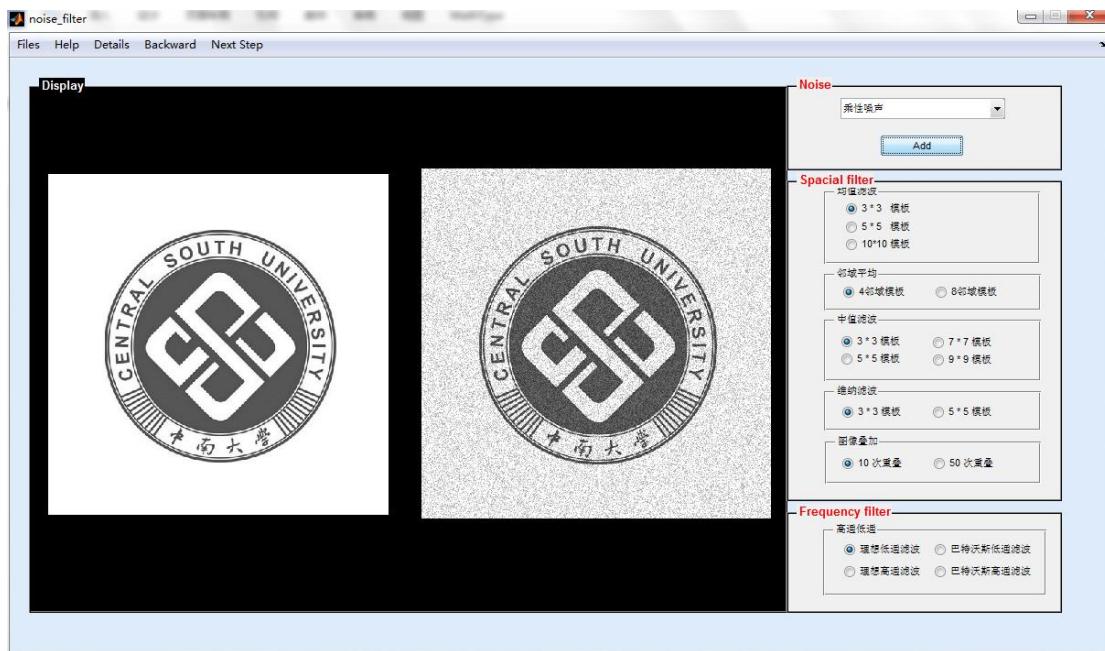
```
cot=0;
while cot<20
cot=cot+1;
for i=1:c
    v(i)=((u(i,:).^2)*x')/sum(u(i,:).^2);
end
disp(v);
for i=1:c
    sumd=zeros(1,m*n);
    for k=1:c
        sumd=sumd+((x-v(i))./(x-v(k))).^2;
    end
    u(i,:)=1./sumd;
end
sumj=0;
for k=1:c
    sumj=sumj+(u(k,:).^2)*((x-v(k)).^2)';
end
J=sumj;
if minJ>J
    minJ=J;
    vmin=v;
    umin=u;
end
end
u=umin;
v=vmin;
```

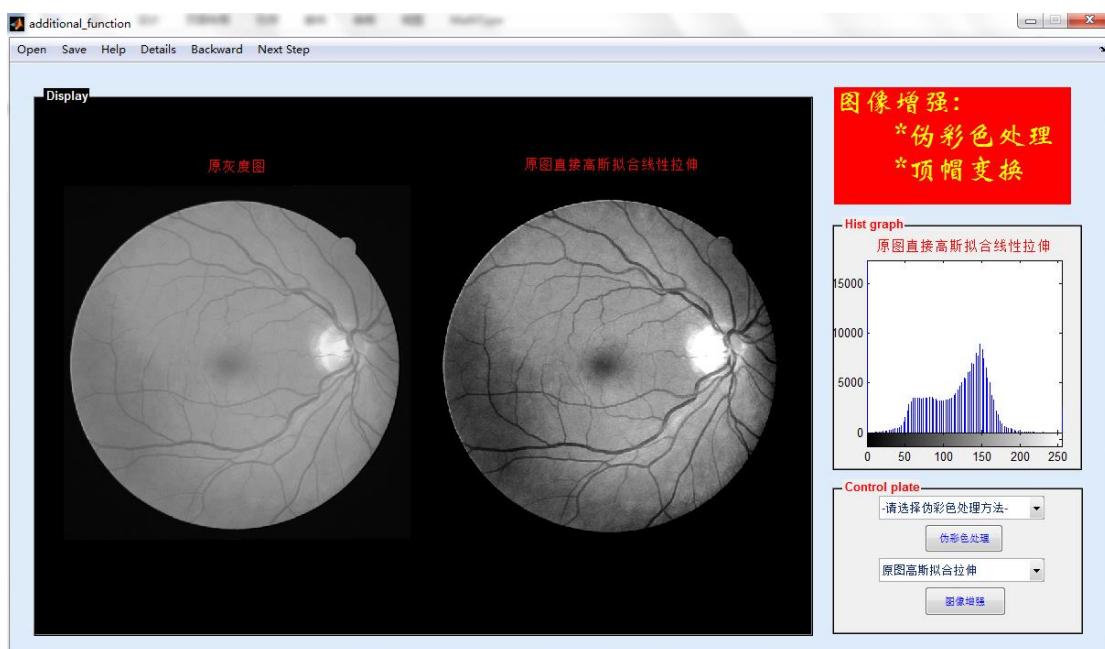
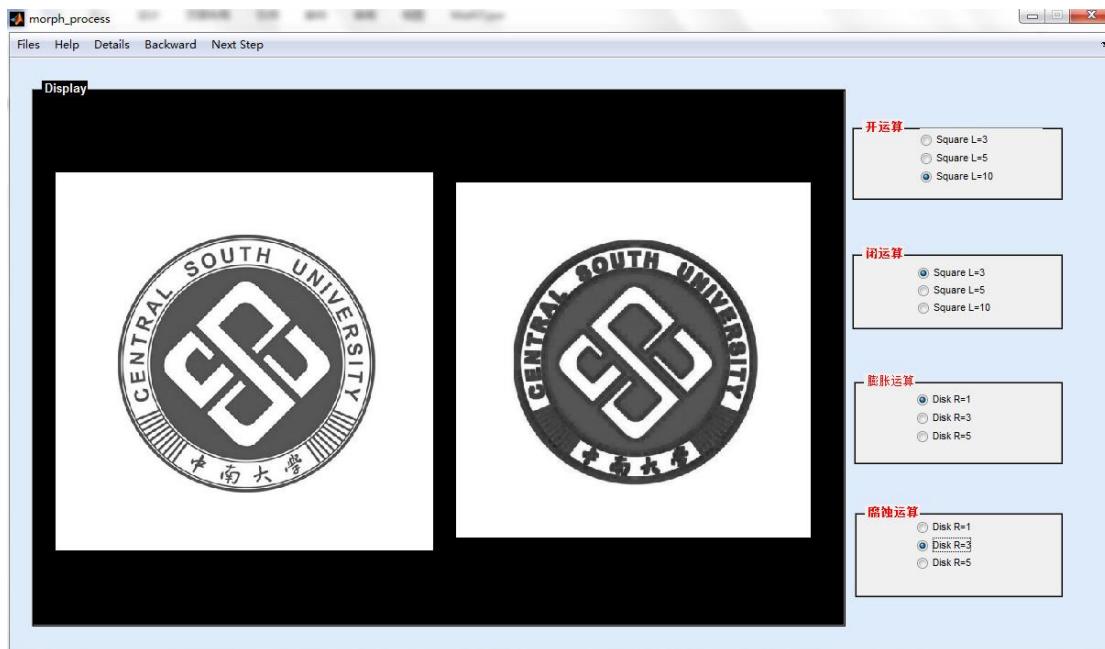


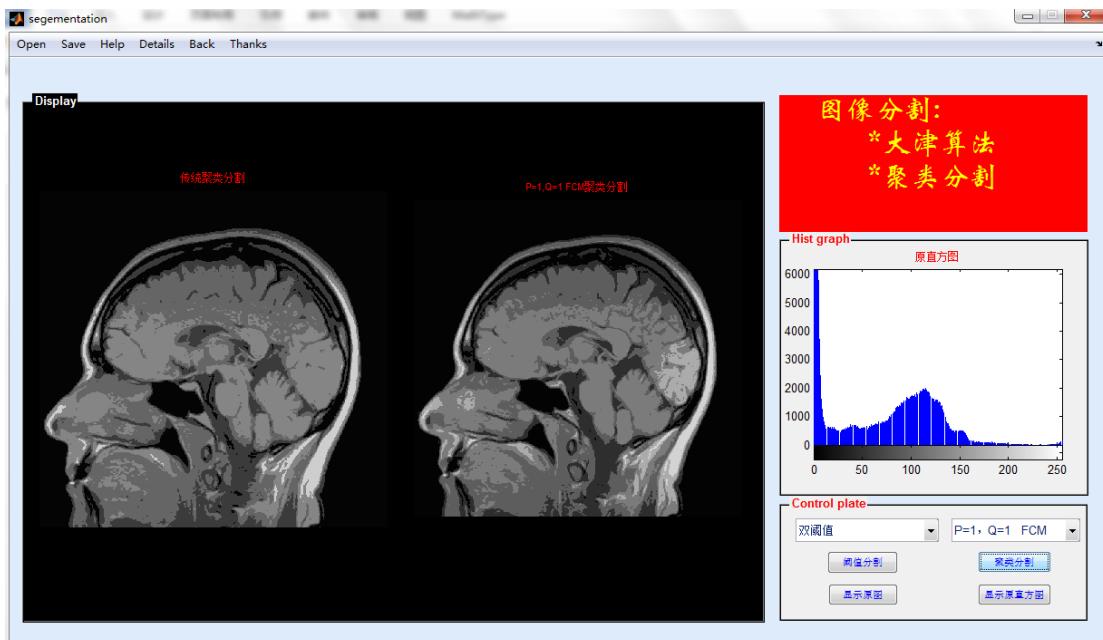


附：界面截图：









六、 参考资料

[1] Digital Image Processing, (Third Edition) , Rafael C. Gonzalez, Richard E. Woods;

[2] 数字图像处理 (第二版) , 冈萨雷斯, 电子工业出版社