实验五 图像分割实验

实验目的

- 掌握梯度边缘检测算子，了解拉普拉斯边缘检测算子和 Canny 边缘检测算子；
- 掌握边界跟踪方法及其原理；
- 掌握自动阈值法和分水岭法。

实验内容

1. 实现三种梯度算子 (Roberts 、 Sobel 、 Prewitt) 的边缘检测
2. 实现对一张二值图像的边界跟踪
3. 实现分水岭算法

实验代码

1.

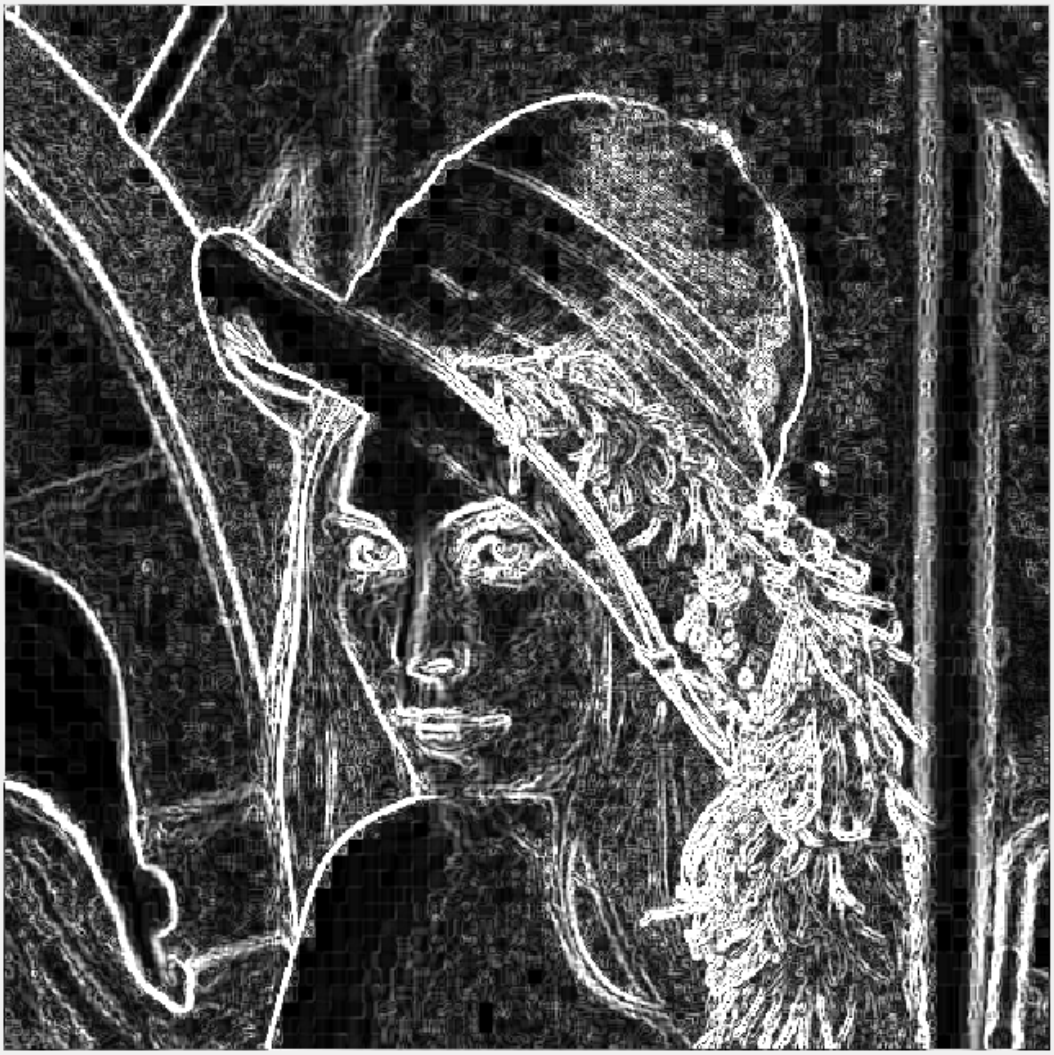
```
# Roberts算子边缘检测
function I =Roberts(I,T)
I=mat2gray(I);  % 实现图像矩阵的归一化操作
[m,n]=size(I);
M=I; % 为保留图像的边缘一个像素
robertsNum=0;  % 经 roberts 算子计算得到的每个像素的值
robertThreshold=T;  % 设定阈值
    for  j=1:m-1  % 进行边界提取
        for  k=1:n-1
            robertsNum = abs(I(j,k)-I(j+1,k+1)) + abs(I(j+1,k)-I(j,k+1));
            if (robertsNum > robertThreshold)
                M(j,k)=255;
            else
                M(j,k)=0;
            end
        end
    end
end

# Sobel边缘检测
function I =Sobeldet(I,T)
I=mat2gray(I); % 实现图像矩阵的归一化操作
[m,n]=size(I);
M=I; % 为保留图像的边缘一个像素
sobelNum=0; % 经 sobel 算子计算得到的每个像素的值
sobelThreshold=T; % 设定阈值
for  j=2:m-1  % 进行边界提取
    for  k=2:n-1
```

```
        sobelNum=abs(I(j-1,k+1)+2*I(j,k+1)+I(j+1,k+1)-I(j-1,k-1)-2*I(j,k-1)-I(j+1,k-1))+abs(I(j-1,k-1)+2*I(j-1,k)+I(j-1,k+1)-I(j+1,k-1)-2*I(j+1,k)-I(j+1,k+1));
         if (sobelNum > sobelThreshold)
         M(j,k)=255;
         else
         M(j,k)=0;
         end
    end
end

end

# Prewwit边缘检测
function I =Prewitterdet(I,T)
I=mat2gray(I); % 实现图像矩阵的归一化操作
[m,n]=size(I);
M=I; % 为保留图像的边缘一个像素
PrewittNum=0; % 经 Prewitt 算子计算得到的每个像素的值
PrewittThreshold=T; % 设定阈值
for  j=2:m-1  % 进行边界提取
  for  k=2:n-1
 PrewittNum=abs(I(j-1,k+1)-I(j+1,k+1)+I(j-1,k)-I(j+1,k)+I(j-1,k-1)-I(j+1,k-1))+abs(I(j-1,k+1)+I(j,k+1)+I(j+1,k+1)-I(j-1,k-1)-I(j,k-1)-I(j+1 ,k-1));
  if (PrewittNum > PrewittThreshold)
 M(j,k)=255;
  else
 M(j,k)=0;
  end
  end
end

end

# 二值追踪
[m,n]=size(img);

imgn=zeros(m,n);  % 边界标记图像
ed=[-1 -1;0 -1;1 -1;1 0;1 1;0 1;-1 1;-1 0];  % 从左上角像素判断
for  i=2:m-1
  for  j=2:n-1
  if  img(i,j)==1  % 如果当前像素为 1

  for  k=1:8
 ii=i+ed(k,1);
 jj=j+ed(k,2);
  if  img(ii,jj)==0  % 当前像素周围如果是背景，边界标记图像相应像素标记
 imgn(ii,jj)=1;
  end
  end

  end
  end
end
img=imgn;
axes(handles.axes1);
imshow(img);
title('二值追踪')
```

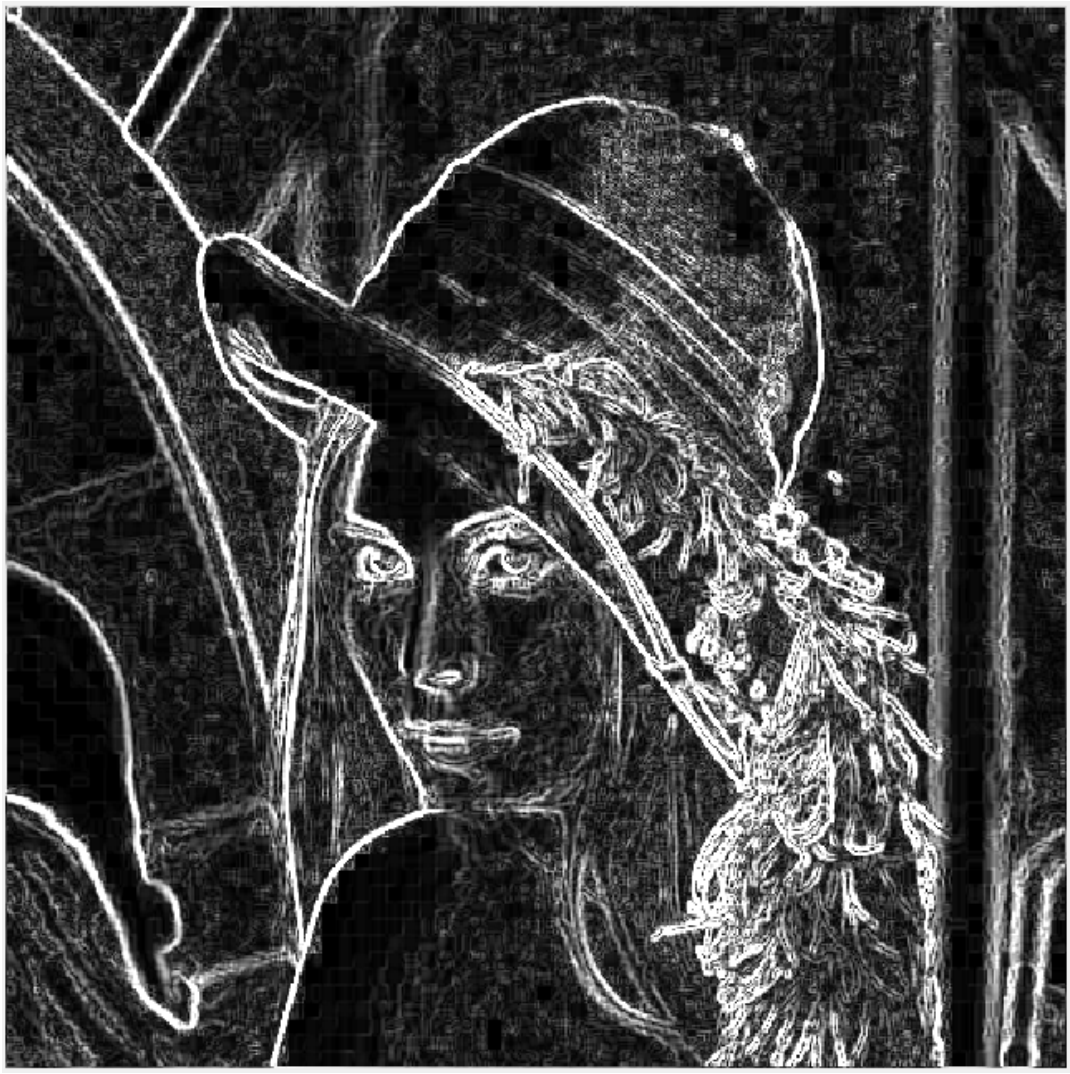
```
84
85  # 分水岭算法实现
86  f=rgb2gray(img);
87  f=double(f);
88  hv=fspecial( 'prewitt' );  % 建立一个预定义的滤波算子
89  hh=hv';  % 计算梯度图
90  gv=abs(imfilter(f,hv, 'replicate' ));
91  gh=abs(imfilter(f,hh, 'replicate' ));
92  g=sqrt(gv.^2+gh.^2);  % 计算距离
93  L=watershed(g);
94  wr=L==0;
95  img=wr;
96  axes(handles.axes1);
97  imshow(img);
98  title( ' 分水岭 ' );
```

实验效果

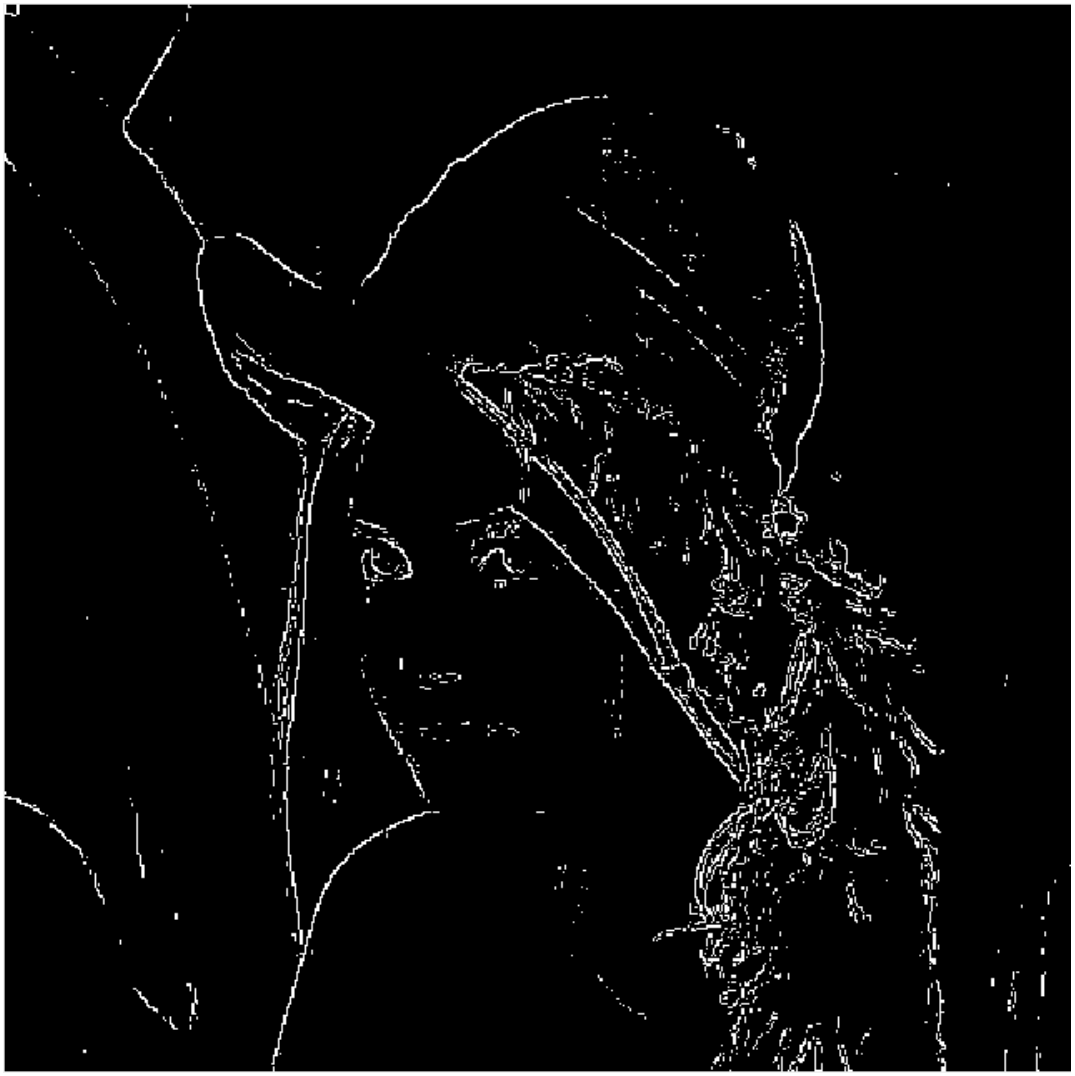
- sobel算子



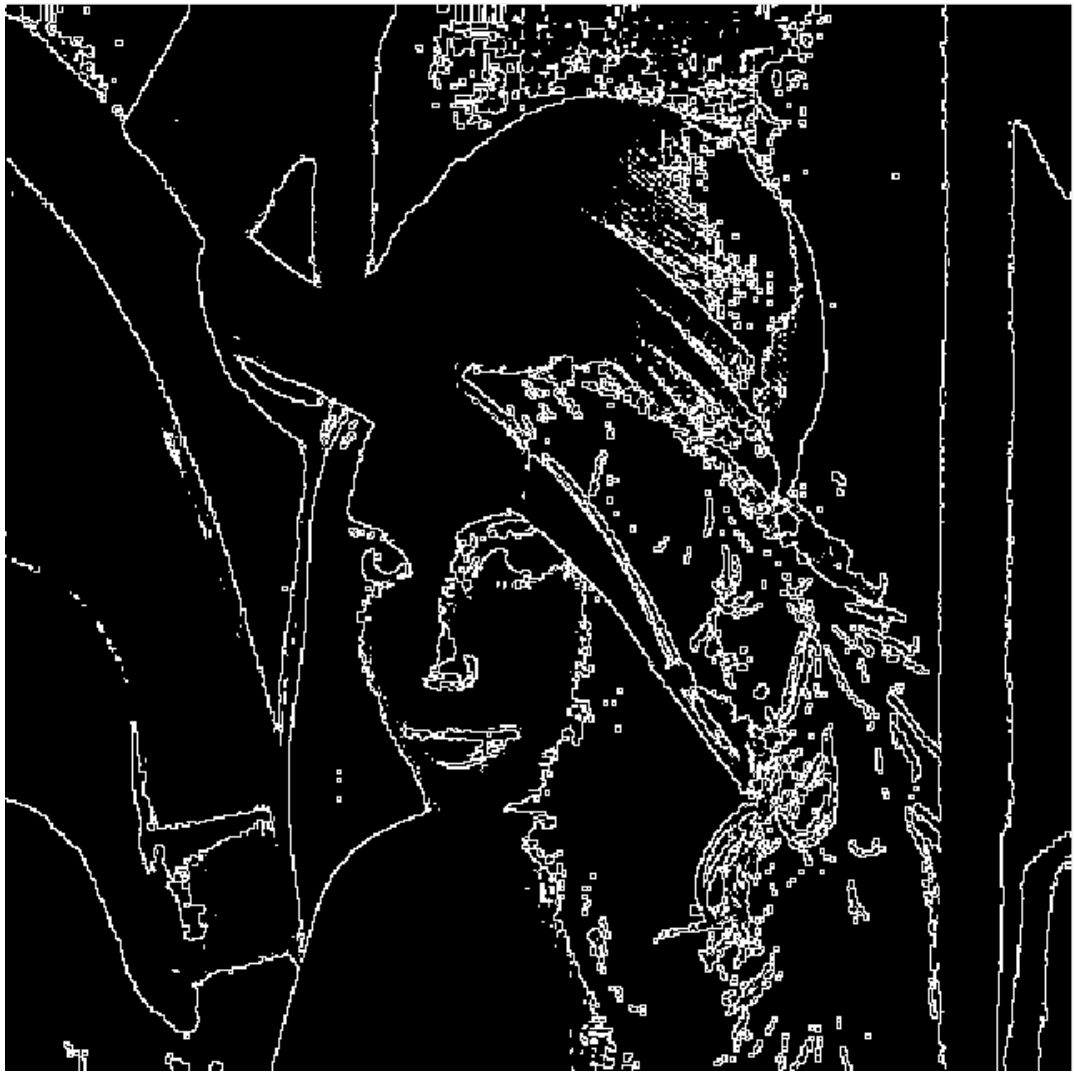
- prewitt算子



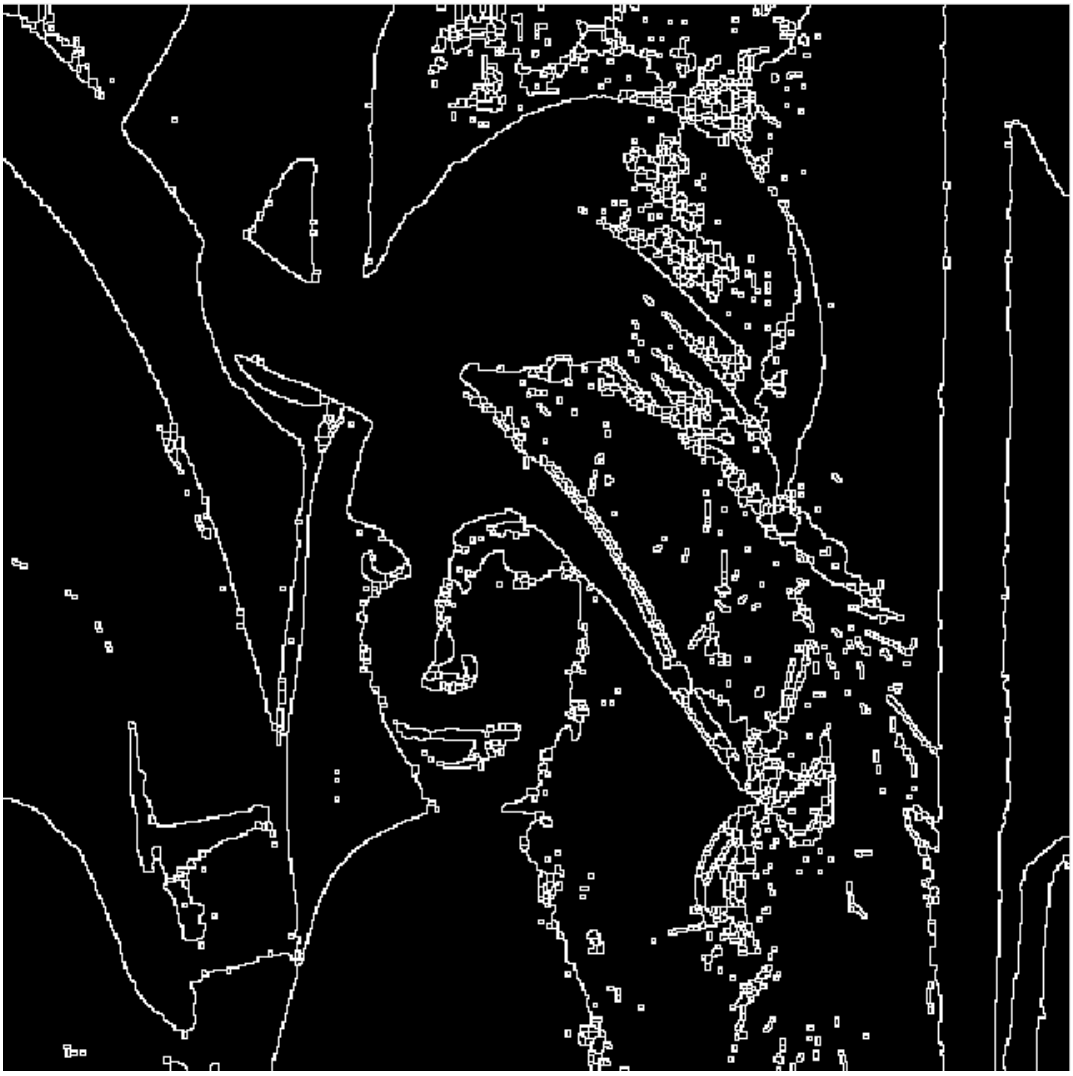
- roberts算子



- 二值边界



- 分水岭



分析思考

1. 彩色图像的边缘检测可以将图像分割成三个不同RGB空间分别检测后合成，但是检测效果不明显，建议转二值化后分割。
2. 分水岭算法容易出现过分割现象。