电子科技大学

University of Electronic Science and Technology of China

实验报告

实验课程： 光电图像处理

姓　　名： 李　宁

学　　号： 2016050201017

实验地点： 211 楼 909

指导老师： 张　静

实验时间： 2018 年 11 月 21 日

一、实验名称：图像分割及目标检测

二、实验目的

1、了解图像边缘检测及图像区域分割的目的、 意义和手段。
2、熟悉各种经典的边缘检测算子、 图像分割方法及其基本原理。
3、熟悉各种图像特征表示与描述的方法及基本原理。
4、熟练掌握利用 MATLAB 工具实现各种边缘检测的代码实现。
5、熟练掌握利用 MATLAB 工具实现基本阈值分割的代码实现。
6、通过编程和仿真实验，进一步理解图像边缘检测、 图像分割及其在目标检测、目标识别及跟踪测量应用中的重要性。

三、实验原理

1、利用 Sobel 算子进行图像的边缘检测

1)、两个方向梯度分量的合成原则。

$$G_1(x,y)=\sqrt{\left(\Delta_x f(x,y)\right)^2+\left(\Delta_y f(x,y)\right)^2}$$

$$G_2(x,y)=\left|\Delta_x f(x,y)\right|+\left|\Delta_y f(x,y)\right|$$

$$G_3(x,y)=\max\left(\left|\Delta_x f(x,y)\right|,\ \left|\Delta_y f(x,y)\right|\right)$$

2)、对梯度进行二值化得到的边缘图。

$$G(x,y)=\begin{cases}1, & if \quad G(x,y)>T\\ 0, & else\end{cases}$$

其中，T 为给定的阈值；最后得到的结果即为二值化的边缘图。

3)、Sobel 算子

$$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$$ x 方向梯度 ， y 方向边缘

$$\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$ y 方向梯度，x 方向边缘

4)、最大类间方差（即 Qtsu 灰度阈值算法）

图像有 L 阶灰度，n_i 是灰度为 i 的像素数，图像总像素数

灰度为 i 的像素概率：$p_i = n_i/N$

类间方差：

$$\sigma_B^2(k)=\omega_1(\mu_1-\mu)^2+\omega_2(\mu_2-\mu)^2$$

$$\mu_1 = \frac{1}{\omega_1}\sum_{i=0}^{k} ip_i, \mu_2 = \frac{1}{\omega_2}\sum_{i=k+1}^{L-1} ip_i, \mu = \sum_{i=0}^{L-1} ip_i$$

$$\omega_1 = \sum_{i=0}^{k} p_i,\ \omega_2 = \sum_{i=k+1}^{L-1} p_i = 1-\omega_1$$

灰度图像阈值：

$$T = \arg\max_{k=1}^{L} \sigma_B^2(k)$$

2、利用梯度改进全局阈值分割

1)、差分形式

$$\Delta_x f(x,y) = f(x,y) - f(x-1,y)$$
$$\Delta_y f(x,y) = f(x,y) - f(x,y-1)$$

2)、梯度合成规则

$$G_1(x,y) = \sqrt{\left(\Delta_x f(x,y)\right)^2 + \left(\Delta_y f(x,y)\right)^2}$$

3、数字图像中目标区域测量及计算

1)、形心

$$\begin{cases} x_c = \dfrac{1}{MN}\displaystyle\sum_{y=0}^{N-1}\sum_{x=0}^{M-1} x \\ y_c = \dfrac{1}{MN}\displaystyle\sum_{y=0}^{N-1}\sum_{x=0}^{M-1} y \end{cases}$$

四、实验步骤

1、利用 Sobel 算子进行图像的边缘检测

1)、如图 1 所示， 一幅数字图像 *I* 与 *Sx* 和 *Sy* 分别做滤波（相关）运算后（可选用多种方式，如 conv2， filter2 及 imfilter 等 MATLAB 函数），可以求得 *x*, *y* 两个方向的梯度图像 *Dx*, *Dy*， 然后，可以计算得到原图像的梯度幅度，即:

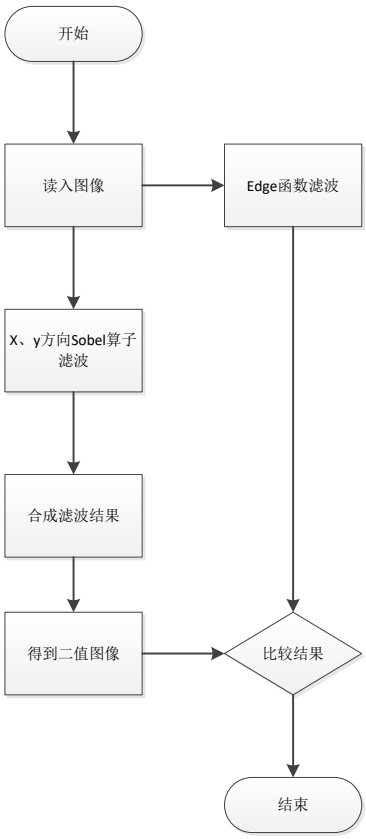
$$D=\sqrt{Dx^2 + Dy^2}$$

2)、进一步执行梯度图像 D 的二值化处理（建议采用 Otsu 阈值，也可考虑其他阈值分割），检测图像的二值化边缘。

3）、对于与步骤同样的输入图像 I，利用 MATLAB 工具的 edge(I,’sobel’)函数进行处理。 试比较处理结果与步骤（2）的得到的结果的差异，并分析检测结果存在一定差异的原因。

4)、画出原图像、原图像的 Dx， Dy， D 图，及最终的边缘检测结果图（即二值化边缘）。

流程图：



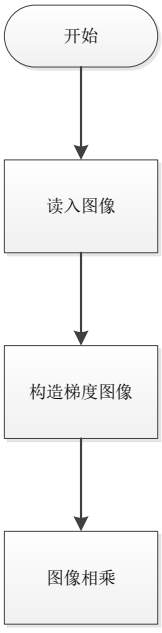
2、利用梯度改进全局阈值分割

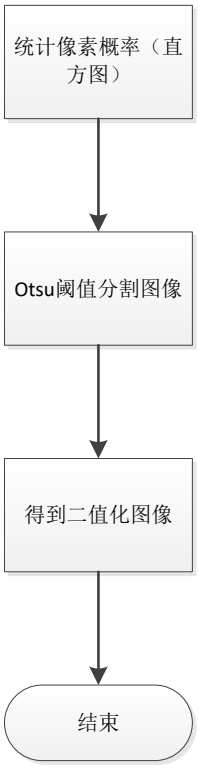
1)、对图 2 中的目标图像利用图像梯度公式，得到梯度幅度图像；

2)、将梯度图像和原图像进行乘积， 得到新的图像；

3)、统计非零像素的直方图；

4)、以统计之后的直方图为基础， 利用 OTSU 阈值处理方法分割乘积后的图像得到最终二值化结果图像。





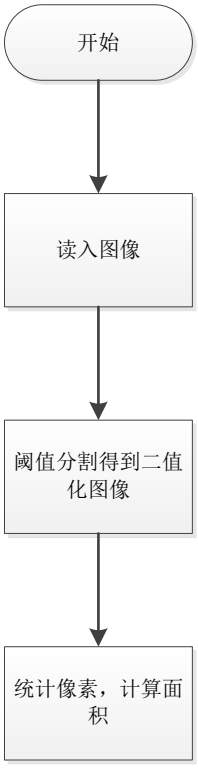
3、数字图像中目标区域测量及计算

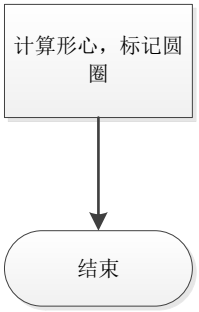
1)、对图 2 中的黑色形状目标进行阈值分割，得到二值化的图像；

2)、计算目标形状的面积（以像素单位表示）；

3)、计算图中黑色形状目标的形心位置，并在原图上进行位置标记（可以计算位置为圆心，以一定半径 r 画一个红色小圆圈）。

其中，A 为目标区面积，f(x,y)在背景区域值为 0，目标区域值为 1。





五、实验结果

1、利用 Sobel 算子进行图像的边缘检测

处理结果：

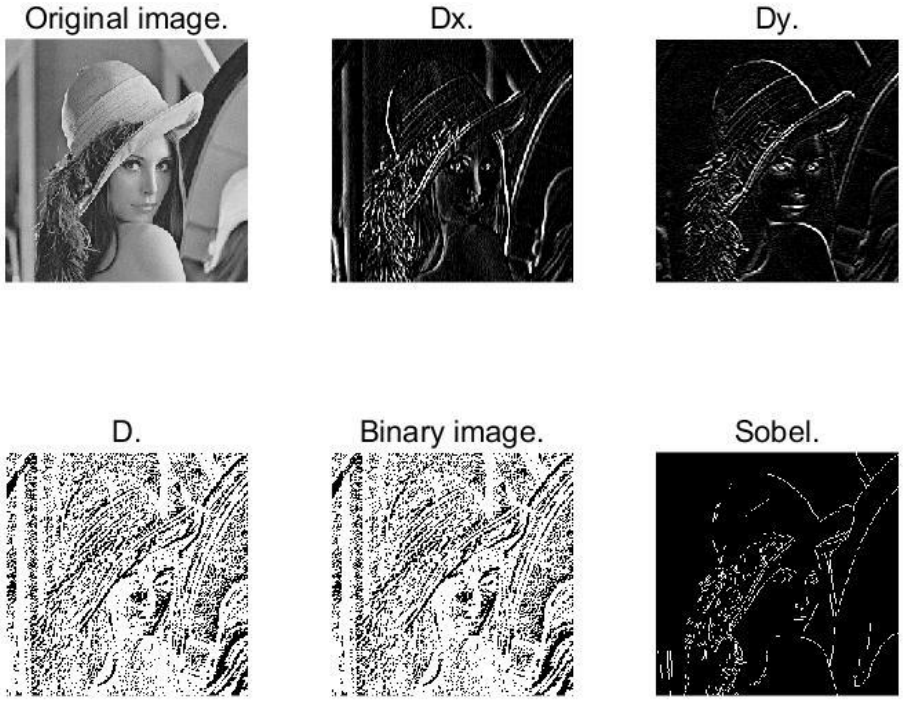


图 1

可以看到通过 Sobel 算子处理后的图像和用 edge 函数处理的图像结果并不相同，我通过查看 D 的具体数值，发现有像素有超过 255 的情况。

解决办法：将 D 进行伸缩变化使 D 值为 0~255 之间，得到 D_u，用 D_u 代替 D 进行二值化处理后得到的处理结果如下：

可以看出最后结果和用 edge 函数处理的结果相似。

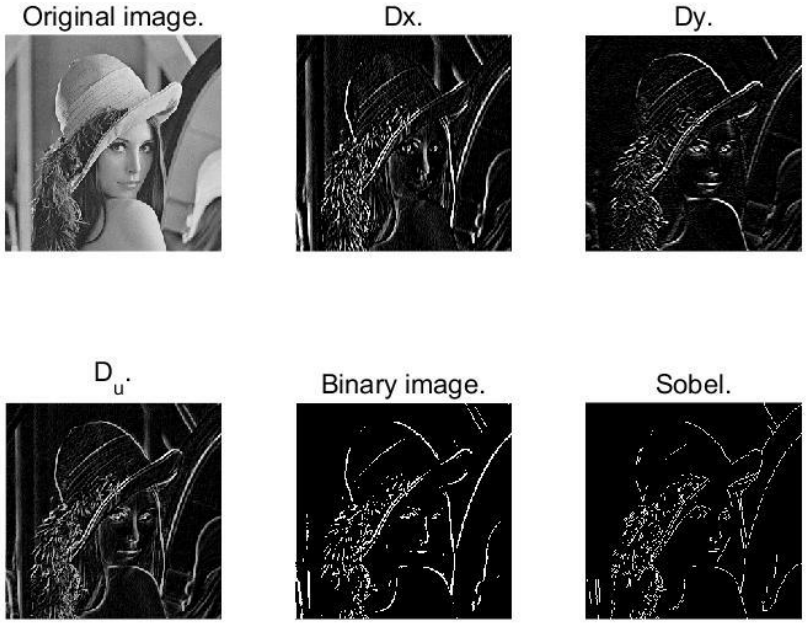


图 2

2、利用梯度改进全局阈值分割

结果：

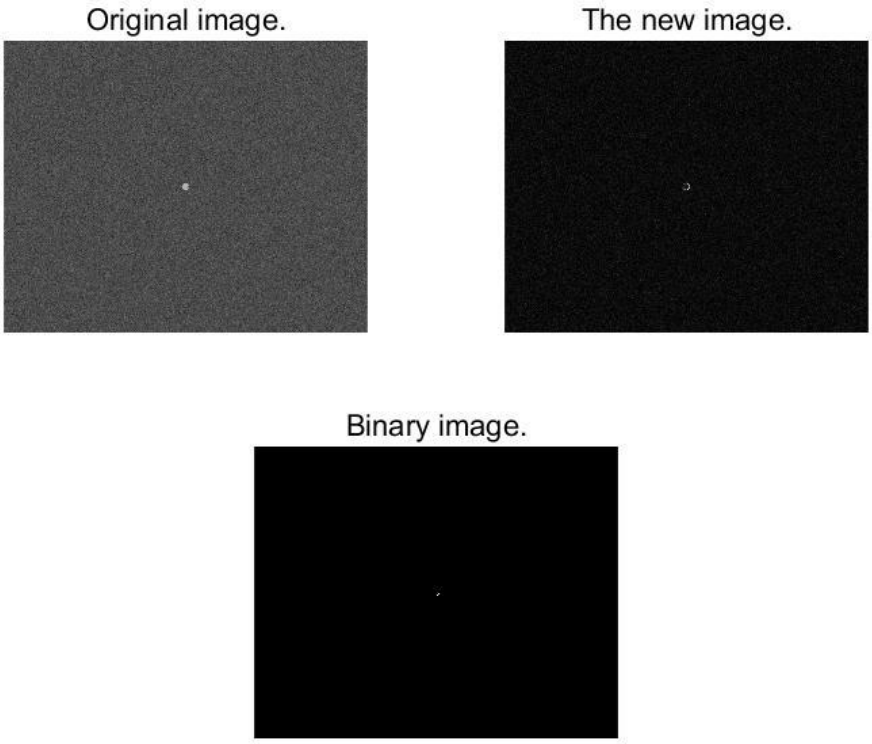


图 3

3、数字图像中目标区域测量及计算

结果：

面积：7846

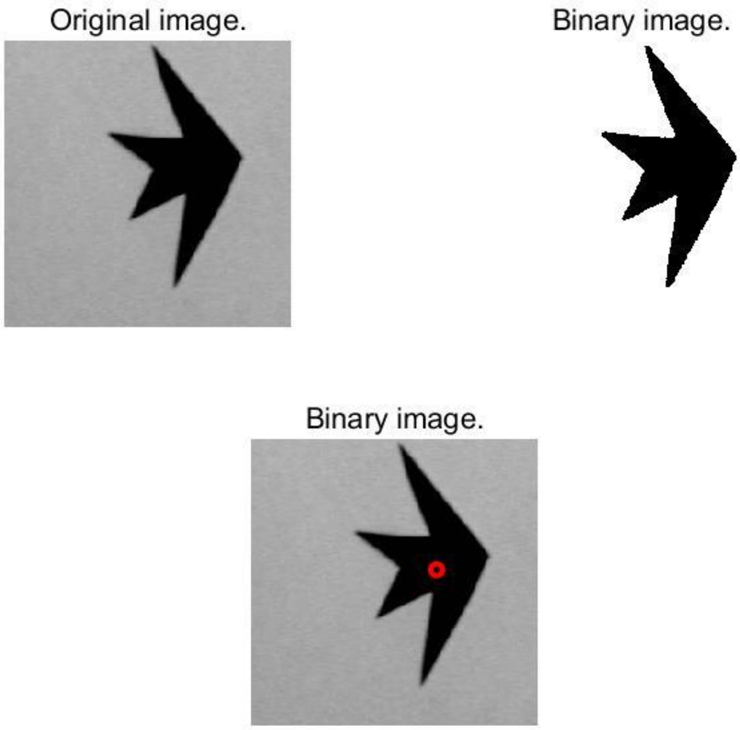


图 4

六、实验心得体会和建议

实验的第二个题，不知为什么统计直方图时为什么要去掉零像素。

七、程序源代码

1、利用 Sobel 算子进行图像的边缘检测

```
clc,clear;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
f0=imread('lena256.jpg');
f=rgb2gray(f0);
[height,width]=size(f);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
Sx=[-1  0  1;
    -2  0  2;
    -1  0  1];
Sy=[-1  -2  -1;
```

```
    0   0   0;
    1   2   1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
Dx=imfilter(f,Sx,'corr','same');
Dy=imfilter(f,Sy,'corr','same');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
Dx_d=double(Dx);
Dy_d=double(Dy);
D=sqrt(Dx_d.^2+Dy_d.^2);
D_u=uint8(mat2gray(D).*255);
level=graythresh(D);
BW=im2bw(D,level);%Binary image.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
level_2=graythresh(D_u);
BW_2=im2bw(D_u,level);%Binary image.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
f_d=double(f);
D_edge=edge(f_d,'sobel','both');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
figure,subplot(2,3,1),imshow(f),title('Original image.');
subplot(2,3,2),imshow(Dx),title('Dx.');
subplot(2,3,3),imshow(Dy),title('Dy.');
subplot(2,3,4),imshow(D),title('D.');
subplot(2,3,5),imshow(BW),title('Binary image.');
subplot(2,3,6),imshow(D_edge),title('Sobel.');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
figure,subplot(2,3,1),imshow(f),title('Original image.');
subplot(2,3,2),imshow(Dx),title('Dx.');
subplot(2,3,3),imshow(Dy),title('Dy.');
subplot(2,3,4),imshow(D_u),title('D_u.');
subplot(2,3,5),imshow(BW_2),title('Binary image.');
subplot(2,3,6),imshow(D_edge),title('Sobel.');
```

2、利用梯度改进全局阈值分割

```
clc,clear;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

```
f0=imread('Fig1042(a)(septagon_small_noisy_mean_0_stdv_10).tif');
f=double(f0);
[height,width]=size(f0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
[P_x,P_y]=gradient(f);
M=sqrt(P_x.^2+P_y.^2);
G=M.*f;
G_u=uint8(mat2gray(G).*255);
x=unique(G_u);
x_1=x(2:size(x));
x_1_d=double(x_1);
y=zeros(size(x_1_d));
for i=1:length(x_1_d)
    y(i)=length(find(f==x(i)));
end
s=height*width-length(find(G_u==0));
frequ=y./s;
T0=1;
temp=0;
for T=1:254
    if find(x==T)
        addr=find(x==T);
        wa=sum(frequ(1:addr));
        wb=1-wa;
        if wa==0||wb==0
            continue;
        else
            ua=(1/wa)*sum(x_1_d(1:addr).*frequ(1:addr));
            ub=(1/wb)*sum(x_1_d(addr+1:end).*frequ(addr+1:end));
            u=wa.*ua+wb.*ub;
            o2=wa.*(ua-u).^2+wb.*(ub-u).^2;
            if o2>temp
                temp=o2;
                T0=T;
            end
        end
    else
        continue;
    end
end
level=T0/255;
BW=im2bw(G_u,level);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%
figure,subplot(2,2,1),imshow(f0),title('Original image.');
subplot(2,2,2),imshow(G_u),title('The new image.');
subplot(2,2,[3,4]),imshow(BW),title('Binary image.');
```

3、数字图像中目标区域测量及计算

```
clc,clear;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
f0=imread('starshape.jpg');
f=double(f0);
[height,width]=size(f0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
level=graythresh(f);
BW=im2bw(f0,0.5);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
s=length(find(BW==0))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
BW_d=double(BW).*255;
g_0=zeros(height,width);
g_1=ones(height,width);
g_0(BW==0)=g_1(BW==0);
y=zeros(height,width);
x=zeros(height,width);
for i=1:height
    for j=1:width
        y(i,j)=i;
        x(i,j)=j;
    end
end
x_c=uint8((1/s).*(sum(sum(x.*g_0))));
y_c=uint8((1/s).*(sum(sum(y.*g_0))));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
figure,subplot(2,2,1),imshow(f0),title('Original image.');
subplot(2,2,2),imshow(BW),title('Binary image.');
subplot(2,2,[3,4]),imshow(f0),...
    rectangle('Position',[x_c,y_c,10,10],'Curvature',[1,1],...
    'EdgeColor','r','LineWidth',2),...
    title('Binary image.');
```

八、思考题

1、 利用梯度算子与图像进行滤波（相关） 运算后，为什么还需要给定阈值进行二值化处理?

便于后续进行图像目标区域显示和分离。

2、 Laplacian 算子检测边缘为什么会产生双边效果？为什么不能检测出边的方向。

Laplacian 算子产生双边效果是由其算法本身的特点决定的，原图像无论是怎样的变化（由黑到白、由白到黑，一阶导数对应白边和黑边），二阶导数均会有亮暗边。

3、相对其他边缘检测算子， Canny 边缘检测算法的主要优势体现在哪里？

Cannny 算子将边缘检测性能优良的三个指标用数学形式表达出来，采用最优化数值方法，既能滤去噪声又能保持边缘特性。