数字图像处理实验二：

- 学号：161220129
- 姓名：王奕琛
- 邮箱：919345923@qq.com
- 时间：2019/4/13

实验思路：

- 边缘检测：分别使用Roberts算子，Laplacion算子，Prewitt算子和Sobel算子进行边缘检测。
  - Roberts算子：一种梯度算子，用交叉的差分表示梯度。算子模板：

    $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ 和 $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$
  - Laplacion算子：一种二阶微分算子，将在边缘处产生一个陡峭的零交叉。算子模板(四邻域)：

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

  - Prewitt算子：一种一阶微分算子，算子模板：

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

  - Sobel算子：离散的一阶差分算子，算子模板：

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- 边缘链接：
  - 由于图像是二值图像，且在之前进行边缘检测时，已将整个图像的像素值划分为V1{0}，V2{255}两个集合，其中像素值为V2中的像素值的点为边界点。
  - 创建一个$[m*2]$的的矩阵M，用来存放进行边缘链接的点
  - 以点(row,col)为起点，判断点(row,col)是否为边界点，若是且未被加入M，则将其加入矩阵M，并对(row,col)的八邻域的点分别进行该操作，若不是则返回上一步操作。
  - 上步操作完成后，M中的点必是相互可达的，对原图中对应于于M中的点置像素为255，其余置像素为0即可。

代码实现：

- 边缘检测：(一般化算法)

```
%不失一般性的边缘检测算法，对不同的算法，其算子计算方式不同，设置的阈值也不同。如下：
```

```
function output = edge(input_image)
  in = input_image;
  [m,n] = size(in);
  output_image = input_image;
  for j = 2:m-1
    for k = 2:n-1
      %robert算子:
      robertsNum = abs(input_image3(j,k)-
input_image3(j+1,k+1))+abs(input_image3(j+1,k)-input_image3(j,k+1));
      %laplacion算子:
      laplacianNum = abs(4*input_image4(j,k)-input_image4(j-1,k)-
input_image4(j+1,k)-input_image4(j,k+1)-input_image4(j,k-1));
      %prewitt算子:
      PrewittNum = abs(in(j-1,k+1)-in(j+1,k+1)+in(j-1,k)-in(j+1,k)+in(j-1,k-
1)-in(j+1,k-1))+abs(in(j-1,k+1)+in(j,k+1)+in(j+1,k+1)-in(j-1,k-1)-in(j,k-1)-
in(j+1,k-1));
      %sobel算子:
      sobelNum=abs(gray(j-1,k+1)+2*gray(j,k+1)+gray(j+1,k+1)-gray(j-1,k-
1)-2*gray(j,k-1)-gray(j+1,k-1))+abs(gray(j-1,k-1)+2*gray(j-1,k)+gray(j-1,k+1)-
gray(j+1,k-1)-2*gray(j+1,k)-gray(j+1,k+1));
      if(Num > Threshold) %Threshold对robert算法和laplace算法取0.2, 对sobel算法取
0.8, 对prewitt算法取0.5
        output_image(j,k) = 255;
      else
        output_image(j,k) = 0;
      end
    end
  end
  figure,imshow(output_image);
  title('kind')
  output = output_image;
end
```

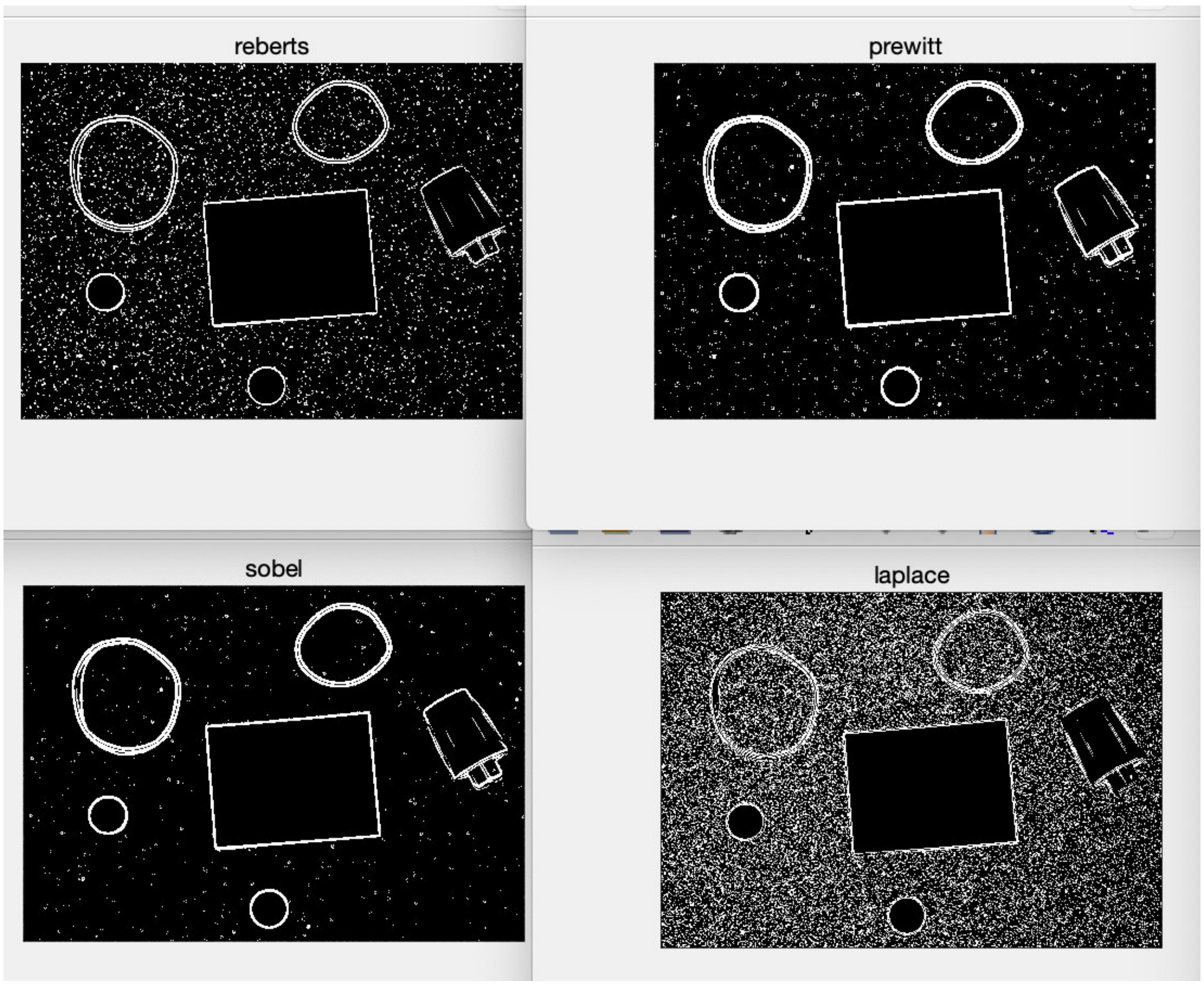
- 边缘链接:

```
function output = my_edgelinking(binary_image,row,col)
  image = binary_image;
  %对应于matlab系统自带的bwtraceboundary函数, 传入点的像素值补能为0;
  if(image(row,col) == 0)
    output = 0;
    return;
  end
  %点(row,col)为M的第一个点
  M = [row,col];
  %将点(row,col)的像素值置0, 避免重复计算
  image(row,col) = 0;
  %对点(row, col)的八邻域的点进行判断
  get_edge(row,col+1);
  get_edge(row+1,col+1);
```

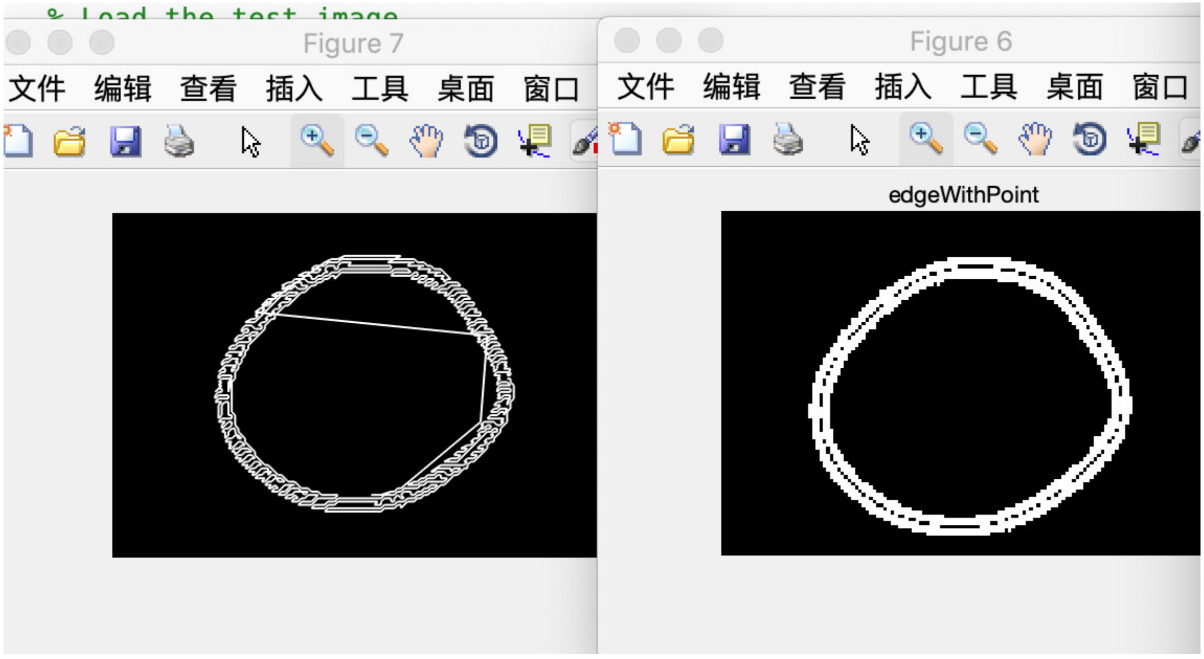
```
    get_edge(row+1,col);
    get_edge(row+1,col-1);
    get_edge(row,col-1);
    get_edge(row-1,col+1);
    get_edge(row-1,col);
    get_edge(row-1,col-1);
    function get_edge(row1,col1)
      %点像素为0则返回，说明该点已被加入M，或不属于边界点
      if(image(row1,col1) == 0)
        return;
      else
        %将该点加入M并置像素值为0;
        M = cat(1,M,[row1,col1])
        image(row1,col1) = 0;
        %对该点的八邻域内的点递归进行函数get_edge;
        get_edge(row1,col1+1);
        get_edge(row1+1,col1+1);
        get_edge(row1+1,col1);
        get_edge(row1+1,col1-1);
        get_edge(row1,col1-1);
        get_edge(row1-1,col1+1);
        get_edge(row1-1,col1);
        get_edge(row1-1,col1-1);
      end
    end
    %按点显示边缘链接的结果
    function show_edge_with_point(b_image)
      [m,n] = size(b_image);
      N = zeros(m,n);
      [p,q] = size(M);
      for i = 1:p
        N(M(i,1),M(i,2)) = 255;
      end
      figure,imshow(N);
      title('edgewithpoint')
     end
     show_edge_with_point(image);
     output = M;
end
```

处理结果:

- 几种边缘检测对比。其中sobel算法和prewitt算法处理得到的效果最好，roberts算法所得结果略受噪声影响，laplace算法所得结果受噪声影响严重。这是正常情况，由laplace算法性质所导致，一般而言我们在使用laplace算法进行边缘检测时，应先对图像进行平滑处理。由此，也可将平滑算法的算子和Laplacion算子结合生成新的模板。

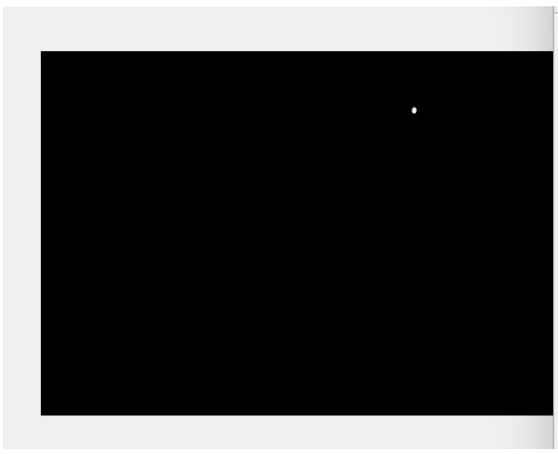


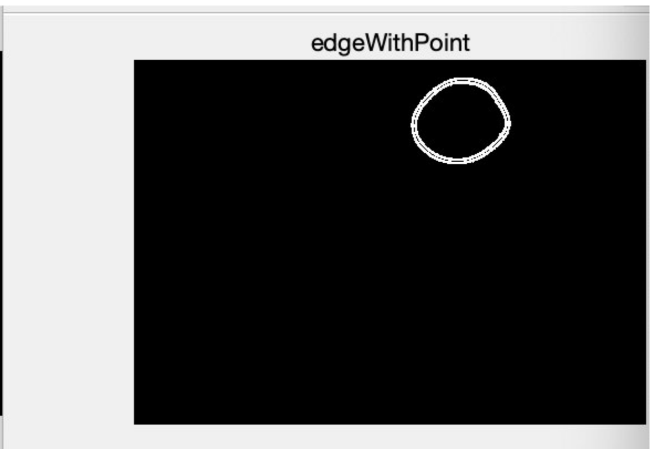
- 边缘链接结果。下图是对rubberand_cap.png图像进行边缘链接的结果，其中F7为使用matlab自带的plot函数所得结果，而F6为对边缘链接所得的矩阵M中的点进行描点所得到的图像。由于矩阵M中的点是按递归顺序所排，存在两个点相距较远的情况，导致使用plot函数所得图像中会有多余的线段。但是通过描点图像可以看出M矩阵中的点均为边界点且符合条件。又由于我们使用的是对八邻域进行判定，从而使我们的边缘链接效果更好。(例如对点[54，339]，使用系统函数bwtraceboundary所得到的仅有一个点，但事实上该点在一个圆的边界上。使用我们的函数可以保证该点所在的圆的边界均被链接。如下图2)



```
  % Load the test image
```

Figure 7
文件 编辑 查看 插入 工具 桌面 窗口
Figure 6
文件 编辑 查看 插入 工具 桌面 窗口
edgeWithPoint




edgeWithPoint