数字图像处理实验二

161220135-吴德亚

本次实验实现了**sobel算子**（同Roberts算子，Prewitt算子类似，故仅实现一种）和较为复杂的**canny算子**，对图像进行了边缘检测，也实现了边缘连接算法：**内边界跟踪算法。**实验过程及报告如下：

一：边缘检测函数

1. **sobel 算子**

   首先设置卷积因子

```
M = [-1,0,1;-2,0,2;-1,0,1];
N = [1,2,1;0,0,0;-1,-2,-1];
```

   对水平和垂直方向的像素点做卷积：（调用imfilter函数，其中'replicate'参数表明：图像大小通过复制外边界的值扩展）

```
img_x = imfilter(input,M,'replicate');
img_y= imfilter(input,N,'replicate');
```

图像的每一个像素的横向及纵向灰度值通过以下公式结合，来计算该点灰度的大小：

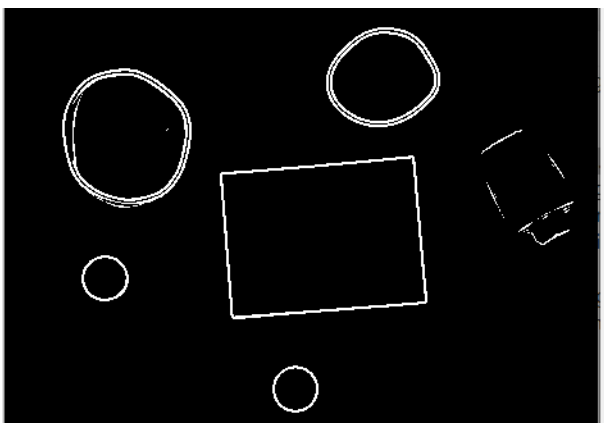
$$\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$$

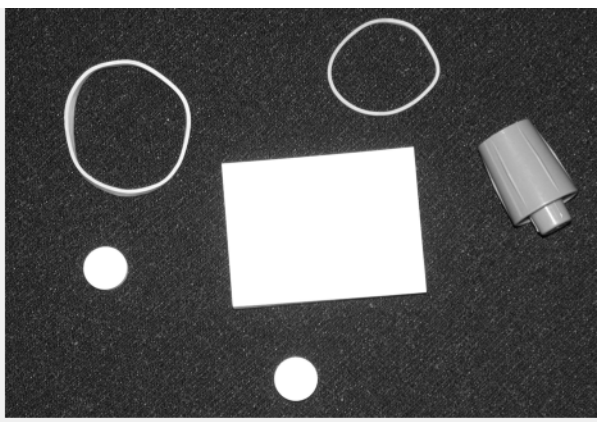
通过遍历图像上的所有像素，调用以上公式计算该点的灰度大小（本实验中256个灰度级映射到0-1区间）

```
t = sqrt(img_x(i,j)*img_x(i,j)+img_y(i,j)*img_y(i,j));
```

为soble算子设计 阈值(threshold)，若该点灰度值大于阈值，保留并置为1，否则舍去置为0.

（在给定的测试样例：rubberband_cap.png， threshold = 1.3)





cloud.jpg， threshold = 0.15

beauty.jpg，threshold = 0.3

2. **canny 算子**

主要分为几个步骤：

- 高斯滤波平滑处理图像

```
gausFilter = fspecial('gaussian',[3,3],sigma); % 高斯低通滤波，标准差为1
output = imfilter(input,gausFilter,'replicate');
```

- 用一阶偏导的有限差分计算梯度的幅值和方向

幅值的计算类比sobel算子即可。

方向则可以通过atand（）函数获取角度，再通过对角度范围判定得到。

```
        Gx = (img_x(i,j))^2;
        Gy = (img_y(i,j))^2;
        p(i,j) = sqrt(Gx+Gy);
        theta = abs(atand(Gx/Gy));  % 得到°的表示
        if ((theta > 22.5) && (theta < 67.5)) % 获取每个点的梯度方向（共分为4个方向）
            sector(i,j) = 0;
        elseif ((theta > 67.5) && (theta < 112.5))
             sector(i,j) = 1;
        elseif((theta > 112.5)&&(theta < 157.5))
            sector(i,j) = 2;
        else
             sector(i,j) = 3;
        end
```

- 对梯度幅值进行非极大值抑制

  该步是canny算法进行边缘检测的重要步骤，是指寻找像素点局部最大值，将非极大值对应的灰度值置为0，这样可以剔除一大部分的非边缘点。

  首先将点M(x,y)同沿着梯度方向的两个8领域像素的梯度值M(x+-1,y+-1)进行比较，如果它是最大的，则保留，否则，表明它不是极大值，需要被删除。

```
switch sector(i,j)  % 对某一点所属区域进行判断
                case 0
                    if ((p(i,j)>p(i-1,j+1))&& (p(i,j)>p(i+1,j-1)))
                        q(i,j) = p(i,j); % 若该点梯度值最大，则保留
                    else
                        q(i,j) = 0;      % 否则置为0
                    end
```

- 用双阈值算法检测

  在次对非极大值抑制后的图像进行阈值检测，如果某一点的梯度值大于higher_bound，则将该点的灰度值置为1，若小于lower_bound，则置为0，若位于两者之间，则去该点8邻域中的最大值t，若t大于higher_bound，则将该点置为1，否则置为0。

```
    for i = 2:img_length-2
        for j = 2:img_width-2
            if(q(i,j) > higher_bound) % 大于higher_bound，置为1
                output(i,j) = 1;
            elseif(q(i,j) < lower_bound) % 小于lower_bound，置为0
                output(i,j) = 0;
            else                           % 找出8邻域中的最大值，与higher,lower bound比较
                t = max([q(i-1,j-1),q(i-1,j),q(i-1,j+1), ...
                    q(i,j-1),q(i,j+1),q(i+1,j-1),q(i+1,j),q(i+1,j+1)]);
                if(t > higher_bound)
                    output(i,j) = 1;
                else
                    output(i,j) = 0;
```

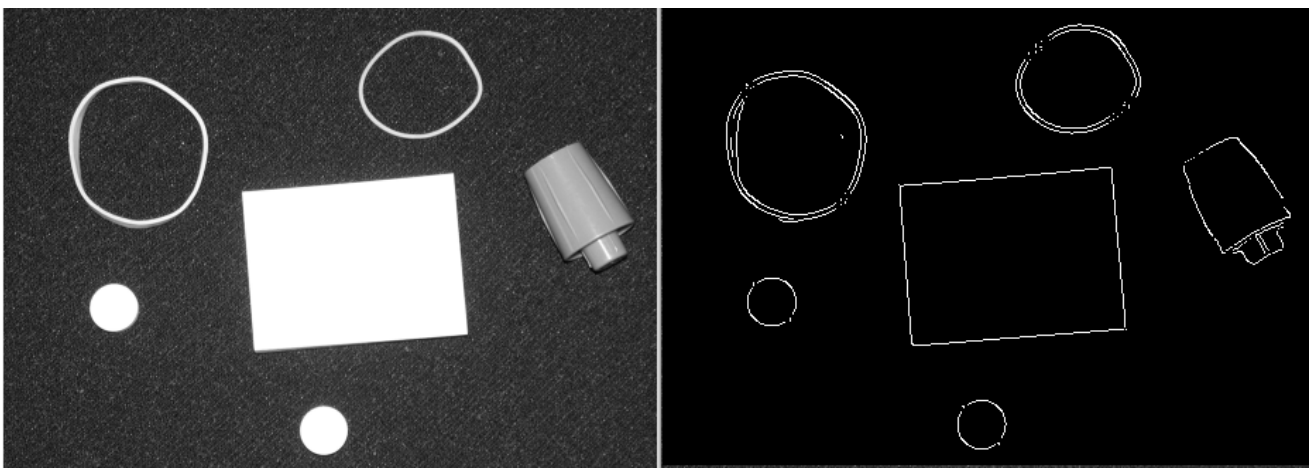
```
                end

            end
        end
    end
```

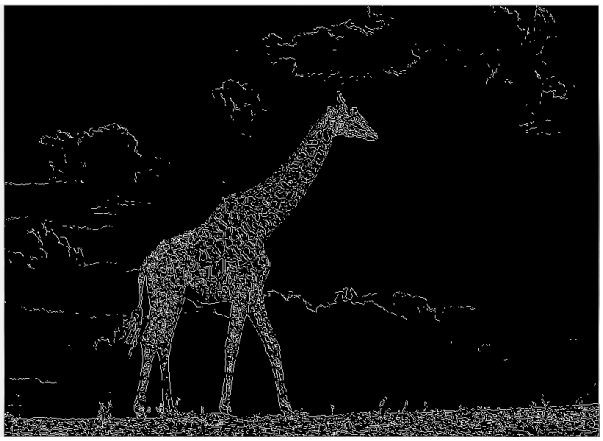
实验效果：

beauty.jpg (lower_bound = 0.1, higher_bound = 0.25;)

rubberband_cap.png (lower_bound = 0.5, higher_bound = 0.6)



giraffe.png (lower_bound = 0.15, higher_bound = 0.2)

二：边缘连接：内边界跟踪算法

- 仅对 rubberband_cap.png 图像进行边缘连接 ， 且由于使用canny算子无法形成连续的边缘，故使用的是sobel算子处理后得到的图像。
- 内边界跟踪算法可以按照4或者8邻域检测，本次实验使用的是8邻接。

  算法实现：

  1. 初始化变量dir=7，设定方向：沿x正半轴为0，沿逆时针方向，每隔45°加一，则方向7出现在 -45°处。
  2. 按照逆时针方向搜索当前3*3邻域，若dir 为偶数，dir = (dir+7)mod8, 若dir为奇数，dir=(dir+6)mod8, 找到的第一个与当前像素值相同（为1）的边界元素Pn, 更新dir的值。
  3. 直到边界元素Pn 与第一个点相重合。

     整体思路如下：

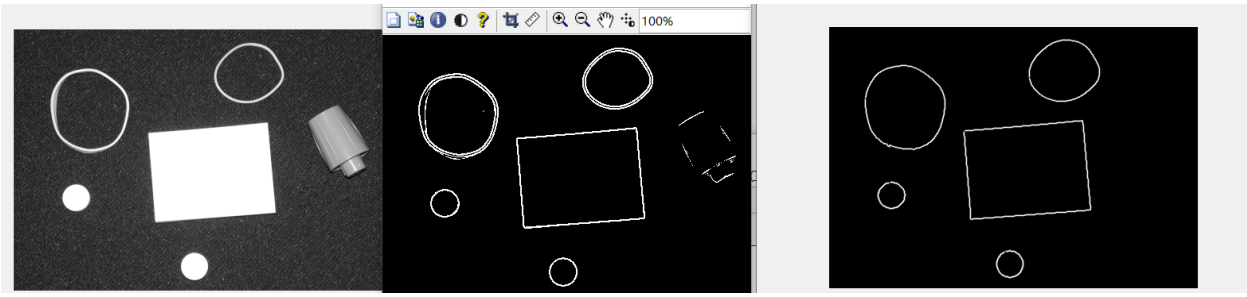
```
next_dot = safe_dot(binary_image,row,col,dir); % 按算法查找下一个点
while(next_dot(1) == 1)      %下一个点存在
    store_dots(numcount,:) = [row,col]; %将点存储在store_dots中，最后返回
    row = next_dot(2);                % 返回值next_dot中存储了新的行列值，dir
    col = next_dot(3);
    dir = next_dot(4);                    %依次更新行列值，dir
    if ((first_dot(1) == row)&&(first_dot(2) == col)) %若与初始点重合，则结束
        output = store_dots(1:numcount,:);  %返回存储的点
        break;
    end
    next_dot = safe_dot(binary_image,row,col,dir);
    % 通过自定义函数safe_dot获取下一步的点，并得到该点的行列值，新的dir
    % safe_dot 函数中包含了判断dir奇偶更新dir,逆时针查找下一个符合条件的点等
    numcount = numcount+1;
end
if(next_dot(1) == 0) %若找不到下一个点，则表明失败
    output = [0,0];
end
```

     实验效果：

     通过imtool工具可以获取边缘的点，通过访问每个图形上一个点的方式来连接该图形。

     存储点： Dots = [131,171;65,252;91,46;196,77;283,222]

（从左到右依次：原图，sobel算子处理后，边缘连接后）



三：实验参考

图像处理基础知识系列之一：边界跟踪之内边界跟踪算法解释

Canny边缘检测算法原理

[图像处理] Sobel边缘检测算法

图像处理常用边缘检测算子总结