本科生实验报告

题	目：	边缘检测之车道线检测
姓	名：	Matthew_Lyu
学	号：	
专	业：	
日	期：	2023年4月20日

1 实验目的

使用Matlab手动实现图像边缘检测，并完成车道线识别任务

2 实验过程

2.1 边缘检测

- 将图片转换成灰度图
- 对灰度图进行高斯滤波
- 利用*Sobel*算子计算像素梯度
- 对梯度图像进行非极大值抑制
- 阈值滞后处理
- 孤立弱边缘抑制

2.2 车道线识别

- 选取ROI区域生成蒙版图像
- 霍夫变换识别直线边缘
- 根据车道线先验知识筛选直线

3 实验方法

3.1 高斯滤波

高斯滤波是一种常用的图像处理技术，用于去除图像中的噪声和平滑图像。它基于高斯函数的特性，可以在图像中应用一个低通滤波器，通过模糊图像来降低图像中的高频信息。

高斯滤波可以用来平滑图像，去除图像中的噪声和细节，也可以用于图像边缘检测。在图像边缘检测中，高斯滤波器可以用于平滑图像，然后应用*Sobel*、*Prewitt*等算子来检测图像的边缘。

高斯滤波可以通过以下公式来实现：

$$G(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{1}$$

其中，x_0和y_0表示滤波器的中心坐标，σ表示标准差，它是控制高斯函数形状和平滑程度的重要参数。

3.2 *Sobel*算子计算梯度

*Sobel*算子是一种常用的边缘检测算法，用于检测图像中的灰度变化。在图像处理中，梯度通常指的是图像像素值的变化率，表示图像中像素强度的变化情况。*Sobel*算子计算图像在水平和垂直方向上的梯度值，用于检测图像中的边缘。

水平梯度和垂直梯度的*Sobel*算子如下所示：

$$
\begin{aligned}
\mathrm{G_x} &= \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \\
\mathrm{G_y} &= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}
\end{aligned} \tag{2}
$$

3.3 非极大值抑制

非极大值抑制是一种用于抑制边缘检测图像中非极大值点的技术。在进行边缘检测时，一般会产生一个梯度图像，其中每个像素都表示该点处梯度的强度和方向。非极大值抑制可以让边缘更细化、更准确。

非极大值抑制的过程如下：

1. 对于梯度图像中的每个像素，确定其在梯度方向上相邻的两个像素点。
2. 比较当前像素处的梯度值和其在梯度方向上相邻的两个像素点处的梯度值。如果当前像素处的梯度值比相邻的两个像素点处的梯度值都大，则保留当前像素，否则将其抑制。
3. 对整个梯度图像进行遍历，将不符合条件的像素点的梯度值设为0。

3.4 阈值滞后处理

阈值滞后处理通常需要高阈值和低阈值。将梯度图像中的每个像素点的梯度值与高阈值进行比较，如果梯度值大于等于高阈值，则认为该像素点属于边缘点。然后，将其周围8邻域内的像素点的梯度值与低阈值进行比较，如果有像素点的梯度值大于等于低阈值，则也将其标记为边缘点。这样得到的边缘点将被认为是真正的边缘点。

3.5 孤立弱边缘抑制

孤立弱边缘抑制的基本思想是通过对梯度图像进行二值化，将梯度值大于某个阈值的像素点标记为边缘点，然后对于每个标记为边缘点的像素点，检查其周围的像素点是否也是边缘点。如果周围的像素点中只有一个或很少几个像素点是边缘点，那么就将该像素点标记为孤立弱边缘，并将其梯度值置为0。

3.6 蒙版图案

使用蒙版图案可以减少图像中的冗余信息。

3.7 霍夫变换检测直线

霍夫变换是一种用于在图像中检测几何形状（如直线、圆、椭圆等）的技术。

4 实现过程

4.1 边缘检测

1. 识别图像并转换为灰度图

```
% 读取图像
img = imread('./2.jpg');

% 转为灰度图像
gray_img = rgb2gray(img);
```

2. 高斯滤波平滑图像

（图片一滤波器大小为1）

```
% 高斯滤波
gauss_img = imgaussfilt(gray_img, 1);
```

（图片二滤波器大小为2.5）

```
% 高斯滤波
gauss_img = imgaussfilt(gray_img, 2.5);
```

3. *sobel*算子计算图像梯度

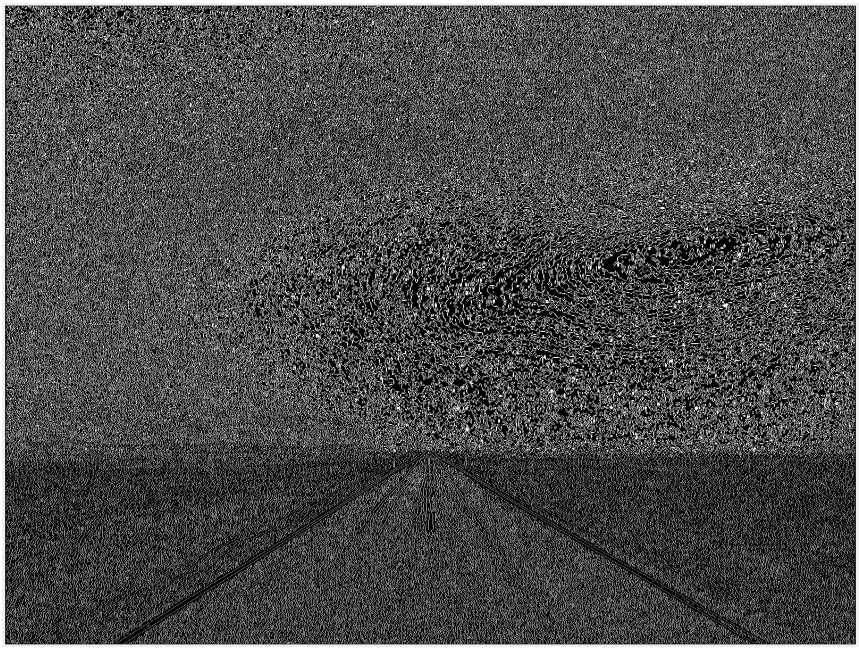
```
% Sobel算子计算像素梯度
sobel_x = [-1 0 1; -2 0 2; -1 0 1];
sobel_y = [-1 -2 -1; 0 0 0; 1 2 1];
grad_x = imfilter(double(gauss_img), sobel_x);
grad_y = imfilter(double(gauss_img), sobel_y);
grad_mag = sqrt(grad_x.^2 + grad_y.^2);
grad_dir = atan2(grad_y, grad_x);
```

4. 非极大值抑制

```
% 非极大值抑制
nms_mag = zeros(size(grad_mag));
for i = 2:size(grad_mag, 1)-1
    for j = 2:size(grad_mag, 2)-1
        dir = grad_dir(i,j);
        mag = grad_mag(i,j);
        % 水平方向
        if (dir > -22.5 && dir <= 22.5) || (dir > 157.5 && dir <= -157.5)
            if (mag >= grad_mag(i,j-1) && mag >= grad_mag(i,j+1))
                nms_mag(i,j) = mag;
            end
        % 45度方向
        elseif (dir > 112.5 && dir <= 157.5) || (dir > -67.5 && dir <= -22.5)
            if (mag >= grad_mag(i-1,j-1) && mag >= grad_mag(i+1,j+1))
                nms_mag(i,j) = mag;
            end
        % 垂直方向
        elseif (dir > 67.5 && dir <= 112.5) || (dir > -112.5 && dir <= -67.5)
            if (mag >= grad_mag(i-1,j) && mag >= grad_mag(i+1,j))
                nms_mag(i,j) = mag;
            end
        % -45度方向
        elseif (dir > 22.5 && dir <= 67.5) || (dir > -157.5 && dir <= -112.5)
            if (mag >= grad_mag(i-1,j+1) && mag >= grad_mag(i+1,j-1))
                nms_mag(i,j) = mag;
            end
        end
    end
end
```

下图是非极大值抑制处理后的梯度图像

（图片一）：



（图片二）：

5. 阈值滞后处理，对于两张图片选取的阈值是不同的

（图片一选取的阈值下限是80，上限是100）

```
% 阈值滞后处理
th_high = 100;
th_low = 80;
edge_img = zeros(size(nms_mag));
edge_img(nms_mag >= th_high) = 1;
edge_img(nms_mag < th_low) = 0;
```

（图片二选取的阈值下限是10，上限是20）

```
% 阈值滞后处理
th_high = 20;
th_low = 10;
edge_img = zeros(size(nms_mag));
edge_img(nms_mag >= th_high) = 1;
edge_img(nms_mag < th_low) = 0;
```

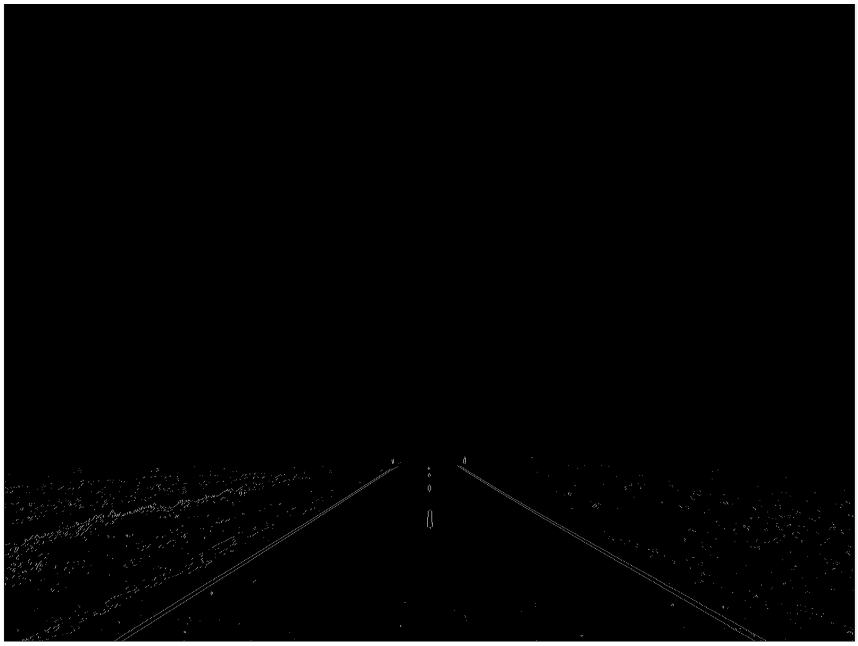
6. 孤立弱边缘处理

由原理可知，孤立弱边缘抑制是检测像素点周围的点是否为边缘，如果有一个是，则将此点也划为强边缘。

```
% 孤立弱边缘抑制
for i = 2:size(nms_mag, 1)-1
    for j = 2:size(nms_mag, 2)-1
        if (nms_mag(i,j) >= th_low && nms_mag(i,j) < th_high)
            if (any(edge_img(i-1:i+1, j-1:j+1) == 1))
                edge_img(i,j) = 1;
            end
        end
    end
end
```

下图展示的是阈值滞后处理和孤立弱边缘处理后的边缘图像

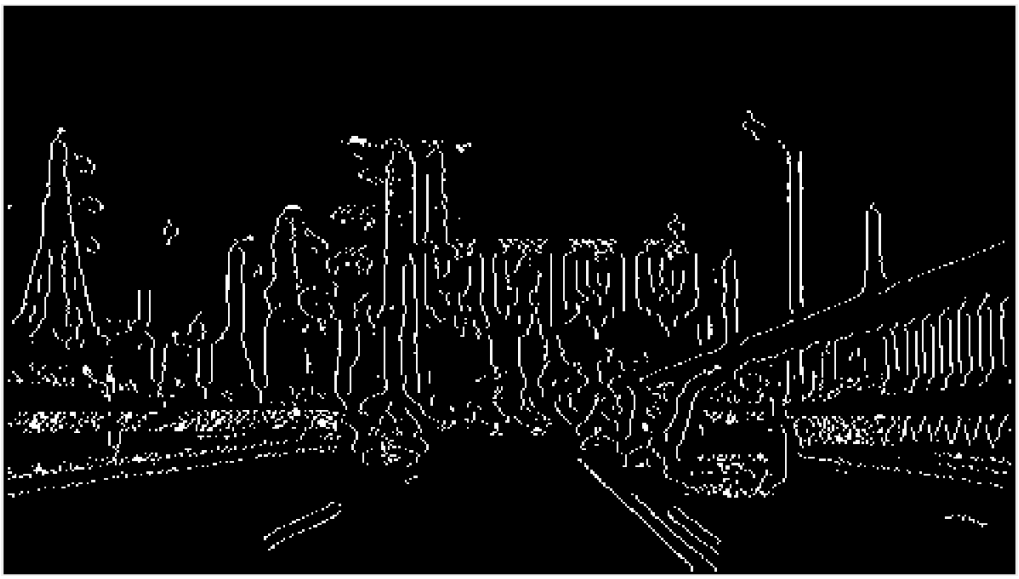
（图片一）：



对于图片二进行孤立弱边缘处理后，使用高斯滤波增强图像

```
% 使用高斯滤波强化图像，使边缘更明显
edge_img = imgaussfilt(edge_img, 0.3);
```

（图片二）：



4.2 车道线检测

1. 生成蒙版图像，这里使用`ginput()`函数划定车道线的ROI区域，然后使用`poly2mask()`函数生成蒙版图像。

```
figure,imshow(edge_img);

[x, y] = ginput(4);
% 选取 ROI 区域生成蒙版图像
roi_mask = poly2mask(x, y, size(edge_img, 1), size(edge_img, 2));
roi_img = edge_img & roi_mask;
```

蒙版图像效果如下：

（图片一）：

（图片二）：

2. 霍夫变换函数检测直线。首先对输入的 ROI 图像进行霍夫变换，得到变换空间中的投票累加器数组 h，以及变换空间中的极角 theta 和极径 rho。然后，通过 houghpeaks 函数，在投票累加器数组 h 中找到最显著的10个峰值，这些峰值表示在变换空间中的一些最有可能对应着图像中的直线。最后，通过 houghlines 函数，在图像中检验可能出现的函数。

```
% 霍夫变换识别直线边缘
[h, theta, rho] = hough(roi_img);
peaks = houghpeaks(h, 10);
lines = houghlines(roi_img, theta, rho, peaks, 'FillGap', 50, 'MinLength', 300);
```

3. 使用先验知识绘制车道线。

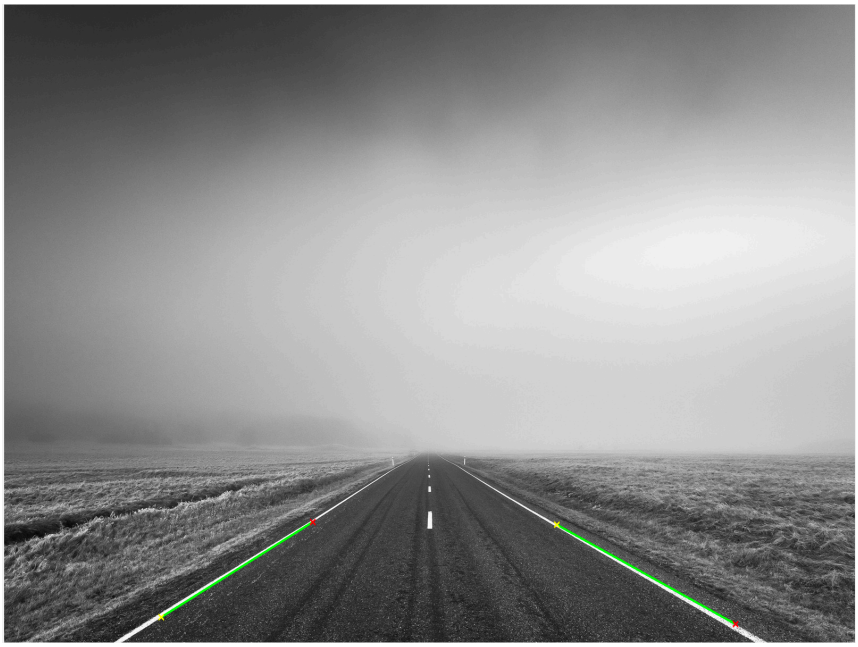
第一张图片的车道线是最长的两条边缘直线，所以这里直接绘制最长的两条直线。

```
% 根据长度排序
lineLengths = zeros(length(lines), 1);
for i = 1:length(lines)
    x1 = lines(i).point1(1);
    y1 = lines(i).point1(2);
    x2 = lines(i).point2(1);
```

```
    y2 = lines(i).point2(2);
    lineLengths(i) = sqrt((x2-x1)^2 + (y2-y1)^2);
end
[sortedLengths, sortedIndices] = sort(lineLengths, 'descend');
sortedLines = lines(sortedIndices);

figure,imshow(gray_img);
hold on
for k = 1:2
   xy =  [sortedLines(k).point1; sortedLines(k).point2];
   plot (xy (:,1), xy (:,2), 'Linewidth' ,2, 'Color', 'green');
   plot (xy (1,1), xy (1,2), 'x', 'Linewidth',2, 'Color','yellow');
   plot (xy (2,1), xy (2,2), 'x', 'Linewidth', 2, 'Color', 'red');
end
hold off
```

以下是第一张图片的检测效果：

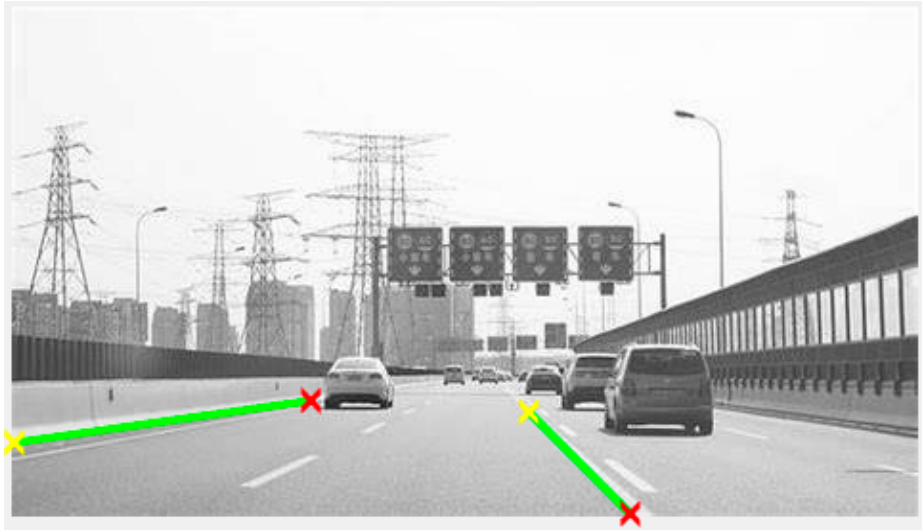


第二张图片同样取识别出的最长的两条直线，但是检测时，需要将距离小于2的直线进行合并：

```
lines = houghlines(roi_img, theta, rho, peaks, 'FillGap', 2, 'MinLength', 5);
```

这里的`FillGap`选择为2

识别效果如下：



5　实验结果与讨论

在本次实验中，我用Matlab实现了图像边缘检测，在实验过程中，分别实现了*Sobel*算子计算像素梯度、非极大值抑制、阈值滞后处理、孤立弱边缘抑制、生成蒙版图像和霍夫变换识别直线，最终较好地完成了车道线识别的任务。

在实验过程中需要调整大量参数，这些参数稍有偏差就会对最后的识别结果产生巨大影响，所以手动检测图像边缘和霍夫变换识别直线对调参的要求较高。在以后的工程和任务中，可以根据实际情况来选择恰当的算法来实现边缘检测。