基于数字图像处理的车道线识别研究

课 程 名 称	数值计算方法
学 院	数据科学与计算机学院
专 业	软件工程（移动信息工程）
年 级 班 级	15 级 20 班
时 间	2017 年 4 月 28 日---5 月 28 日
指 导 老 师	纪庆革

目录

姓名	学号	邮箱
钟思根	15352445	553244295@qq.com
钟诗婷	15352444	1062637471@qq.com
周禅城	15352448	604658513@qq.com

摘要：

为了将数值计算方法课程的理论知识与实际情况相结合，我们研究了车道线识别的算法，首先对图像进行预处理，确定感兴趣区域，高斯滤波，双峰法阈值分割图像，边缘检测，从而减少噪声等无关因素的干扰，提高检测的准确性。接着用霍夫变换对车道线进行识别。为了检测算法的准确性，我们用带确定斜率直线的图片进行检测，将检测出来的直线与原直线进行对比，做误差分析。研究的软件平台为 matlab。

关键词：车道线检测 霍夫变换 曲线拟合 Matlab

1.选题背景：

车道线识别是数字图像处理的一个应用，具有广阔的发展前景，特别是随着智能汽车，自动化的发展，车道线识别发挥着越来越重要的作用，一个好的车道线识别算法不仅能使汽车在驾驶过程中更加顺利，更重要的是对车主的人身安全能够有更好的保障。

车道线识别的研究成果很大程度上得益于数字图象处理和计算机技术的发展。数字图像处理(Digital Image Processing)是通过计算机对图像进行去除噪声、增强、复原、分割、提取特征等处理的方法和技术。

数字图像处理的工具可分为三大类：

第一类包括各种正交变换和图像滤波等方法，其共同点是将图像变换到其它域（如频域）中进行处理（如滤波）后，再变换到原来的空间（域）中。

第二类方法是直接在空间域中处理图像，它包括各种统计方法、微分方法及其它数学方法。

第三类是数学形态学运算，它不同于常用的频域和空域的方法，是建立在积分几何和随机集合论的基础上的运算。

车道线识别现已运用于智能汽车的车道保持辅助系统。车道保持辅助系统属于智能驾驶辅助系统中的一种。它是在车道偏离预警系统（LDWS）的基础上对刹车的控制协调装置进行控制。

对车辆行驶时借助一个摄像头识别行驶车道的标识线将车辆保持在车道上提供支持。

如果车辆接近识别到的标记线并可能脱离行驶车道，那么会通过方向盘的振动，或者是声音来提请驾驶员注意。

如果车道保持辅助系统识别到本车道两侧的标记线，那么系统处于待命状态。

我们希望能够在车道线识别的过程中，适当地用到数值方法课程的内容，比如曲线拟合，

矩阵卷积，误差分析，梯度计算等，加深对课程的理解，在实际操作中运用到课程学到的数值方法的知识，提高自己的实践能力。

图像理解虽然在理论方法研究上已取得不小的进展，但它本身是一个比较难的研究领域，存在不少困难，因人类本身对自己的视觉过程还了解甚少，因此计算机视觉是一个有待人们进一步探索的新领域

2.理论知识

在我们的车道线识别预处理过程中，很重要的一个步骤就是进行边缘检测。edge 函数是 Matlab 软件内建的一个求解图像边缘信息的函数，其提供了多种边缘比较器适配不同的使用场景。常用的边缘比较器有三种，包括**“Sobel”算子、“Canny”算子和“Log”算子**。下面分别介绍三种算子的计算原理。

在介绍各种算子之前，首先要介绍矩阵卷积的概念。

矩阵卷积的计算步骤：

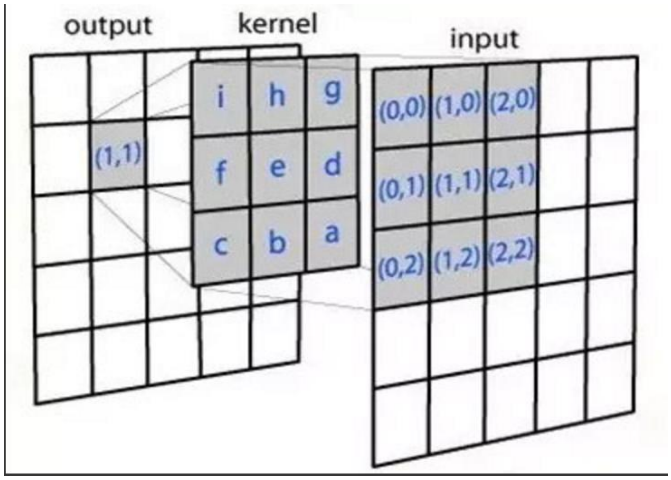
（1）卷积核绕自己的核心元素顺时针旋转 180 度

（2）移动卷积核的中心元素，使它位于输入图像待处理像素的正上方

（3）在旋转后的卷积核中，将输入图像的像素值作为权重相乘

（4） 第三步各结果的和做为该输入像素对应的输出像素

下面是矩阵卷积的图示：

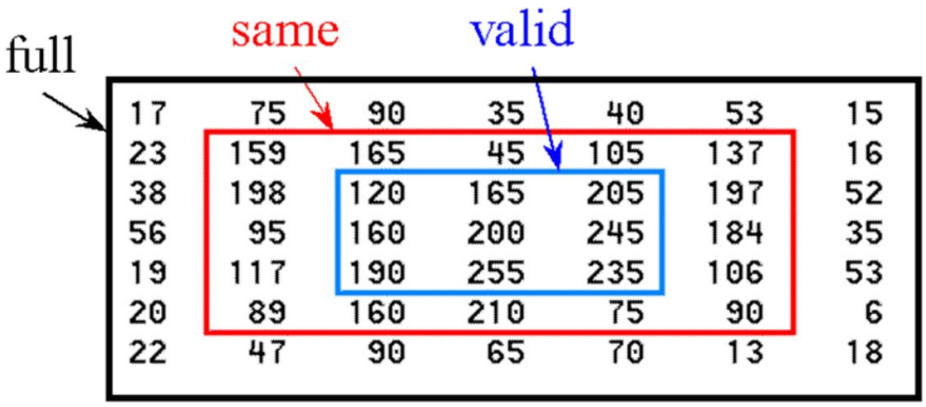


另外对于边缘元素的不同处理有 3 种情况：Valid、Same 和 Full，它们对应的是不同的卷积范围，注意在遇到边缘元素时，如果元素不足就补 0.

以下面的矩阵为例：

$$\begin{pmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{pmatrix} * \begin{pmatrix} 1 & 3 & 1 \\ 0 & 5 & 0 \\ 2 & 1 & 2 \end{pmatrix}$$

对应的卷积结果为：



“Sobel”算子

计算机视觉领域的一种重要处理方法。主要用于获得数字图像的一阶梯度，常见的应用和物理意义是边缘检测。在技术上，它是一个离散的一阶差分算子，用来计算图像亮度函数的一阶梯度之近似值。在图像的任何一点使用此算子，将会产生该点对应的梯度矢量或是其法矢量。

该算子包含两组 3x3 的矩阵，分别为横向及纵向，将之与图像作平面卷积，即可分别得出横向及纵向的亮度差分近似值。如果以 A 代表原始图像，Gx 及 Gy 分别代表经横向及纵向边缘检测的图像，其公式如下:

$$Gx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \qquad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

图像的每一个像素的横向及纵向梯度近似值可用以下的公式结合，来计算梯度的大小：

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

可用以下公式计算梯度方向：

$$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G_y}}{\mathbf{G_x}}\right)$$

“Canny”算子

Canny 边缘检测算子是 John F. Canny 于 1986 年开发出来的一个多级边缘检测算法。其包括 4 个主要步骤：

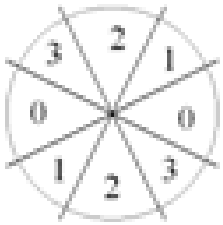
1.消除噪声。一般情况下，使用高斯平滑滤波器卷积降噪。 如下显示了一个 size = 5 的高斯内核示例:

$$K = \frac{1}{139}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2.计算梯度幅值和方向。这个步骤与上述的 Sobel 算子一致，不再赘述。

3.对梯度幅值进行非极大值抑制。仅仅得到全局的梯度并不足以确定边缘，因此为确定边缘。

必须保留局部梯度最大的点，而抑制非极大值。利用梯度的方向进行抑制。



1	2	3
8		4
7	6	5

四个扇区的标号为 0 到 3，对应 3*3 邻域的四种可能组合。在每一点上，邻域的中心象素 M 与沿着梯度线的两个象素相比。如果 M 的梯度值不比沿梯度线的两个相邻象素梯度值大，则令 M=0。

即：$$N[i,j]=NMS(M[i,j],\xi[i,j])$$

4.滞后阈值。最后一步，Canny 使用了滞后阈值，滞后阈值需要两个阈值（高阈值和低阈值）：

a) 如果某一像素位置的幅值超过高阈值，则该像素被保留为边缘像素；

b) 如果某一像素位置的幅值低于低阈值，则该像素被排除；

如果某一像素位置的幅值在两个阈值之间，该像素仅仅在连接到一个高于高阈值的像素是被保留。

“Log”算子

Log 算子来源于 Marr 视觉理论中提出的边缘提取思想,即先对原始图像进行平滑处理,从而实现对噪声最大程度的抑制,再对平滑后的图像提取边缘。LOG 算子被誉为最佳边缘检测算子之一。基本步骤：

（1）采用二维高斯滤波器平滑滤波。二维高斯滤波器的函数 G(x,y)

$$G(x,y)=\frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2})$$

用 G(x,y)与原始图像 f(x,y)卷积，得到平滑图：

$$I(x,y)=G(x,y)*f(x,y)$$

（2）采用二维拉普拉斯算子进行图像增强。拉普拉斯算子的两个卷积核如下所示：

$$G_X=\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix}G_Y=\begin{bmatrix}-1&-1&-1\\-1&-8&-1\\-1&-1&-1\end{bmatrix}$$

或者表示为：$$M(x,y)=\nabla^2\{I(x,y)\}=\nabla^2[G(x,y)*f(x,y)]=[\nabla^2G(x,y)]*f(x,y)$$

求取 M(x,y)的零穿点轨迹即可得到图像 f(x,y)的边缘。以∇^2G(x,y)对原始灰度图像进行卷积运算后提取的零交叉点作为边缘点。

Hough Transform（霍夫变换）

Hough 变换于 1962 年由 Paul Hough 提出，并在美国作为专利被发表。它所实现的是一种从图像空间到参数空间的映射关系。Hough 变换是一种使用表决原理的参数估计技术。其原理是利用图像空间和 Hough 参数空间的点－线对偶性，把图像空间中的检测问题转换到参数空间。通过在参数空间里进行简单的累加统计，然后在 Hough 参数空间寻找累加器峰值的方法检测直线。Hough 变换的实质是将图像空间内具有一定关系的像元进行聚类，寻找能把这些像元用某一解析形式联系起来的参数空间累积对应点。在参数空间不超过二维的情况下，这种变换有着理想的效果。

理解霍夫变换，首先我们从直线坐标参数空间入手：
在图像 x-y 空间中，经过点(x_i , y_i)的直线表示为：

$$y_i=ax_i+b$$

其中，参数 a 为斜率，b 为截距。
通过点(x_i , y_i)的直线有无数条，而且对应不同的 a 和 b 值。如果我们将想 x_i 和 y_i 视为常数，而将原本的参数 a 和 b 看作变量，则上式可以表示为：

$$b=-x_ia+y_i$$

这样就变换到了参数平面 a-b，这个变换就是直角坐标中对于(x_i , y_i)点的霍夫变换。
该直线是图像坐标空间中的点(x_i , y_i)在参数空间的唯一方程。考虑到图像坐标空间中的另一点(x_j , y_j)，它在参数空间中也有相应的一条直线，表示为：

$$b=-x_ja+y_j$$

这条直线与点(x_i , y_i)在参数空间的直线相交于一点(a_0 , b_0)，如图所示：

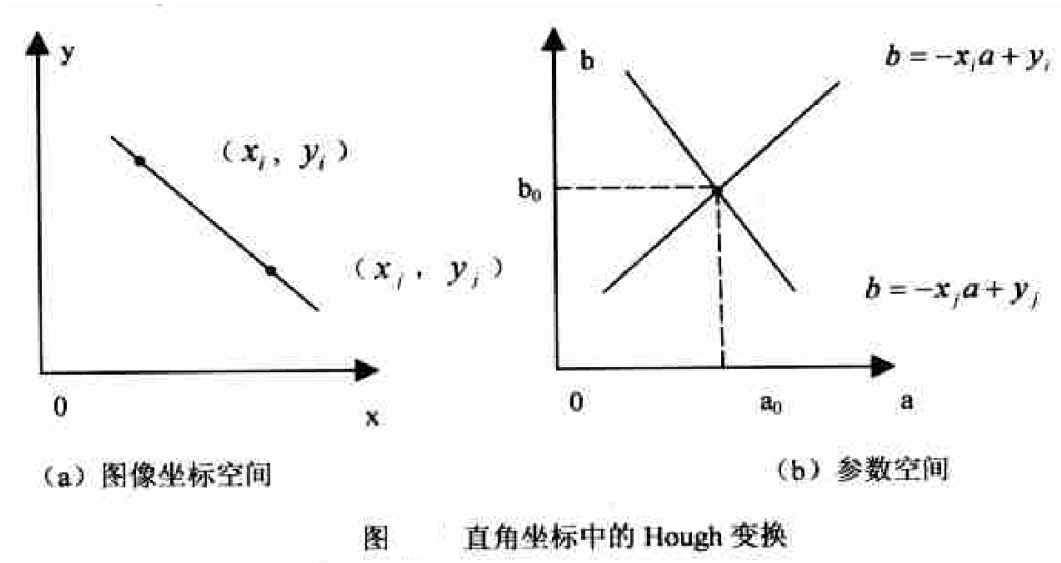


图　　直角坐标中的 Hough 变换

图像坐标空间中过点(xi,yi)和点(xj,yj)的直线上的每一点在参数空间 a－b 上各自对应一条直线，这些直线都相交于点(a0,b0),而 a0、b0 就是图像坐标空间 x－y 中点(xi,yi)和点(xj,yj)所确定的直线的参数。 反之，在参数空间相交于同一点的所有直线，在图像坐标空间都有

共线的点与之对应。根据这个特性，给定图像坐标空间的一些边缘点，就可以通过 Hough 变换确定连接这些点的直线方程。

具体计算时，可以将参数空间视为离散的。建立一个二维累加数组 A(a,b),第一维的范围是图像坐标空间中直线斜率的可能范围，第二维的范围是图像坐标空间中直线截距的可能范围。开始时 A(a,b)初始化为 0，然后对图像坐标空间的每一个前景点(xi,yi),将参数空间中每一个 a 的离散值代入式子(2)中，从而计算出对应的 b 值。每计算出一对(a,b),都将对应的数组元素 A(a,b)加 1，即 A(a,b)=A(a,b)+1。所有的计算结束之后，在参数计算表决结果中找到 A(a,b)的最大峰值，所对应的 a0、b0 就是源图像中共线点数目最多(共 A(a,b)个共线点)的直线方程的参数；接下来可以继续寻找次峰值和第 3 峰值和第 4 峰值等等，它们对应于原图中共线点略少一些的直线。注：由于原图中的直线往往具有一定的宽度，实际上相当于多条参数极其接近的单像素宽直线，往往对应于参数空间中相邻的多个累加器。因此每找到一个当前最大的峰值点后，需要将该点及其附近点清零，以防算法检测出多条极其邻近的“假”直线。

对于上图的 Hough 变换空间情况如下图所示：

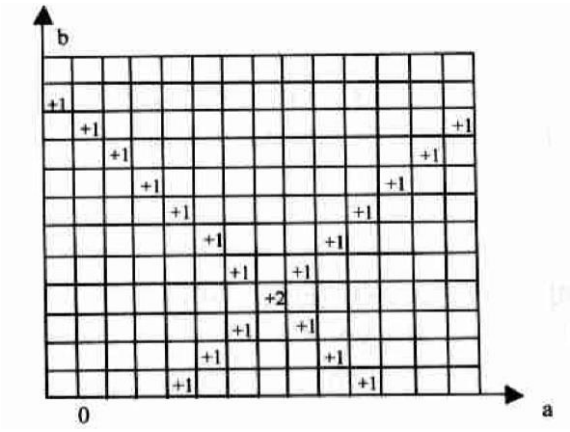


图　参数空间表决结果

实际上由于可能存在直线垂直于 x 轴的情况，因为此时直线斜率为无穷大，无法用一个有限数表示，所以一般采用极坐标参数空间。

极坐标中用如下参数方程表示一条直线：

$$\rho = x\cos\theta + y\sin\theta$$

其中 ρ 代表直线到原点的垂直距离，θ 代表 x 轴到直线垂线的角度，取值范围±90◦，如图所示

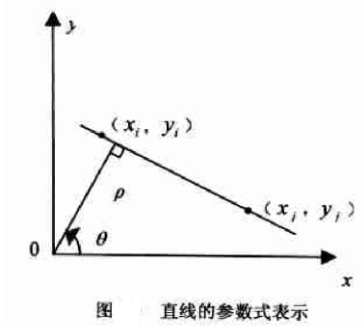


图　直线的参数式表示

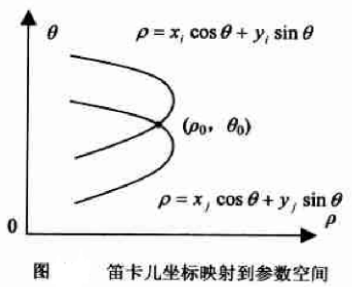


图　笛卡儿坐标映射到参数空间

与直角坐标类似，极坐标中的 Hough 变换也将图像坐标空间中的点变换到参数空间中。

在极坐标表示下，图像坐标空间中共线的点变换到参数空间中后，在参数空间都相交于同一点，此时所得到的 ρ 、 θ 即为所求的直线的极坐标参数。与直角坐标不同的是，用极坐标表示时，图像坐标空间的共线的两点(xi,yi)和(xj,yj)映射到参数空间是两条正弦曲线，相交于点(ρ0,θ0),如上图所示。

具体计算时，与直角坐标类似，也要在参数空间中建立一个二维数组累加器 A，只是取值范围不同。对于一副大小为 D×D 的图像，通常 ρ 的取值范围为[−2 √ D/2,2 √ D/2],θ 的取值范围为[−90◦,90◦]。计算方法与直角坐标系中累加器的计算方法相同，最后得到最大的 A 所对应的(ρ,θ)。

曲线拟合

在后面的实验过程中，我们还尝试使用曲线拟合去完成弯曲车道线的识别。我们在这里主要讨论的是最小二乘法和样条函数插值。

1.最小二乘法

设$\{(x_k, y_k)\}$，其中 k=1:N，其中横坐标$\{x_k\}$是确定的，最小二乘拟合曲线

$$Y=Ax+B$$

的系数是下列线性方程的解，这些方程称为正规方程：

$$\begin{cases} a_0 m + a_1 \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} y_i \\ a_1 \sum_{i=1}^{m} x_i^2 + a_0 \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} x_i y_i \end{cases}$$

2.样条函数插值

最小二乘法求解出来的拟合曲线在某些情况下不能很好地处理极大极小值，而且还有可能会摆动以经过样本点。因此样条函数将图形分段，每一段为一个低阶多项式，并在相邻点之间进行插值。

在实验中我们采用的是**三次紧压样条**，也就是说对于每一个分段都是采用三次函数进行拟合，而端点约束为已知一阶导数在端点取值。

3.实验过程

预处理

识别车道线，第一步就是要进行边缘检测，因此我们分别使用三种不同的算子进行边缘检测观察哪种算子更适用于车道线检测。值得注意的是对于在行驶过程中车辆通过摄像头获取到的道路信息是非常有特点的，首先我们感兴趣的区域集中在图像下半部，图像上

半部分主要是天空等无用信息，其次对于我们感兴趣的区域也是非常有特点的，首先对于道路区域是比较平滑的，没有什么噪声干扰，但是对于道路以外的区域存在绿化带等非常复杂的无用信息干扰。

预实验：使用一幅比较有代表性的车道图作为样例，观察三种算子的边缘检测效果。实验结果如下图。

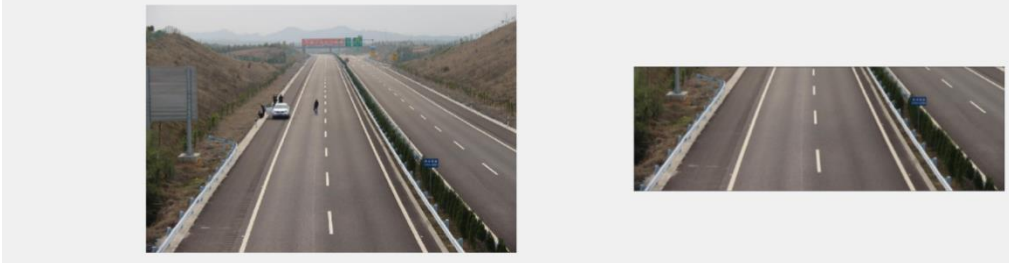
从图中可以看出对于车道线识别中的边缘检测算法，Canny 算子识别的效果最好，因为边缘信息保留较为完全，而其他两种算子边缘信息丢失较多。经查阅文献三种算子分别有以下特点：

1.**sobel 算子**检测方法对灰度渐变和噪声较多的图像处理效果较好，sobel 算子对边缘定位不是很准确，图像的边缘不止一个像素。

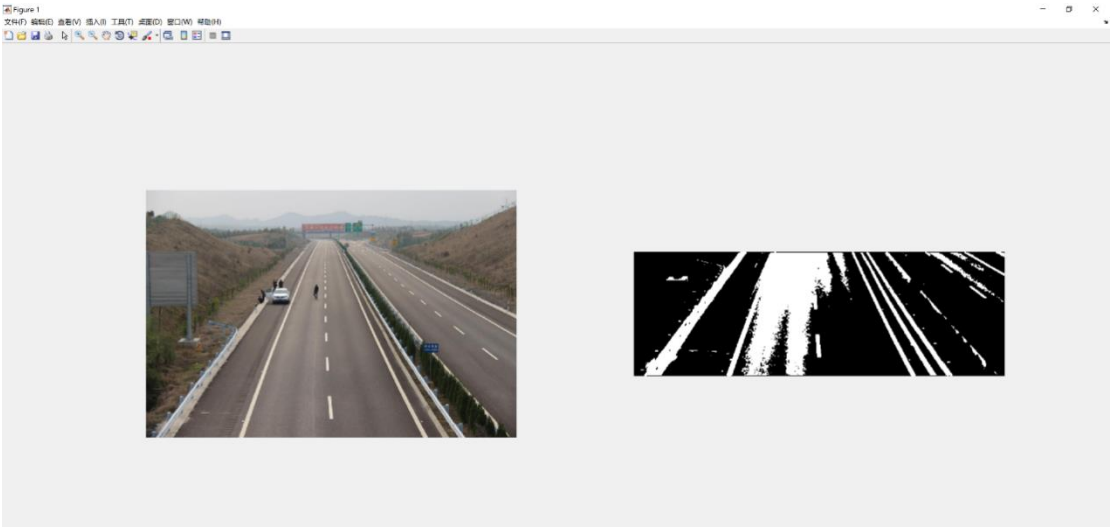
2.**Log 算子**法对噪声比较敏感，所以很少用该算子检测边缘，而是用来判断边缘像素视为与图像的明区还是暗区。

3.**Canny** 方法不容易受噪声干扰，能够检测到真正的弱边缘。优点在于，使用两种不同的阈值分别检测强边缘和弱边缘，并且当弱边缘和强边缘相连时，才将弱边缘包含在输出图像中。

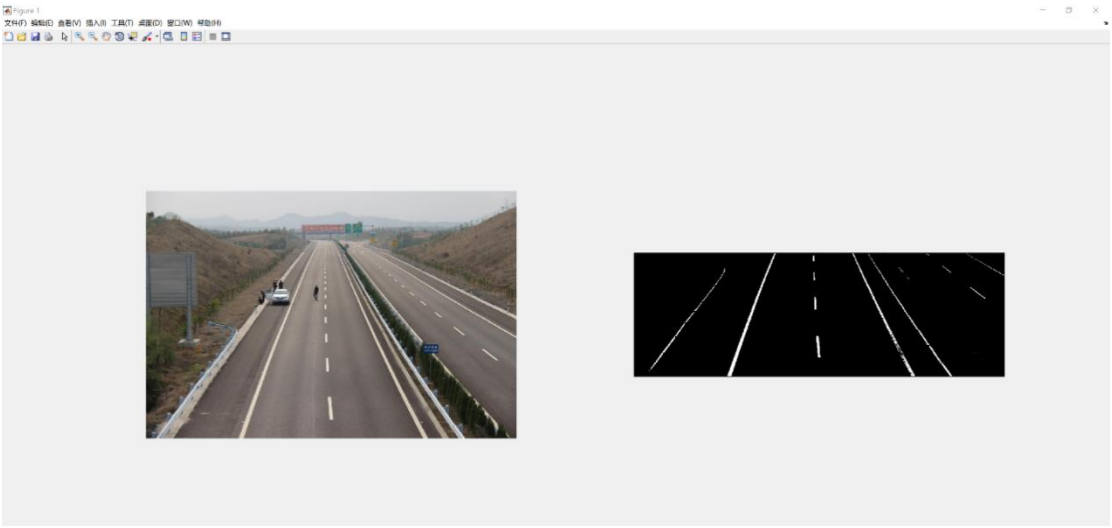
接下来，为了提高检测效率，避免无用噪声的干扰，我们首先将图像上半部分移除，然后再取剩下的图像（后面我们对左侧和右侧车道分开检测，进一步将剩下的图像分成左右两半），然后参考相关文献我们决定利用高斯滤波器进行平滑处理，消除某些噪点。同样是上面的图片，经过上述处理之后我们得到下面的结果。



我们研究小组仔细观察了经过上述处理后得到的图像，经过讨论后一致认为这样的图像对于车道线识别来说还是不合适的，因为图片是彩色的，实际上对于车道线识别图像的彩色信息是无用信息，而且还会增加时间复杂度。然后我们尝试使用 Matlab 内建的将彩色图像转化为灰度图的函数对图像进行处理，发现结果依旧不理想，主要是车道周边的噪声信息干扰依旧非常严重。经过我们长时间的讨论我们终于找到了一个一举多得的办法。首先我们意识到车道线基本上白色的，但是车道基本上都是黑色或者说近似于黑色的灰色。白色和黑色在 RGB 上是非常有对比的，对于每一个颜色分量白色都是 255，黑色是 0。因此我们想到对于任选的一个颜色分量，**我们设置一个阈值**，当高于这个阈值就将这个颜色变为白色，低于这个阈值时就将这个颜色变为黑色。这样一来，既可以去除掉我们不需要的彩色信息，同时还可以去除噪声，凸显我们需要的车道线信息。下面两幅图分别是阈值为 120 和 200 的情况，可以看到当我们将阈值设 200 的时候有非常好的识别效果。



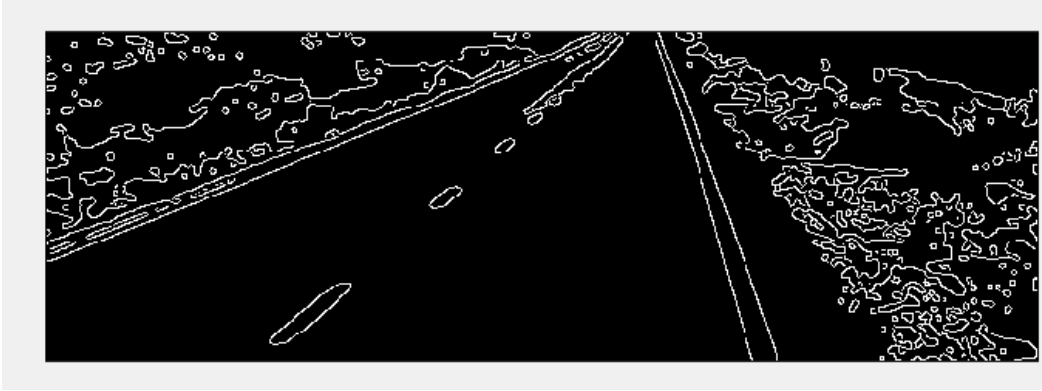
阈值 120



阈值 200

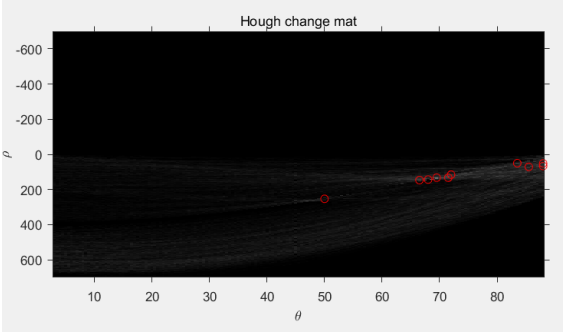
霍夫变换的应用：

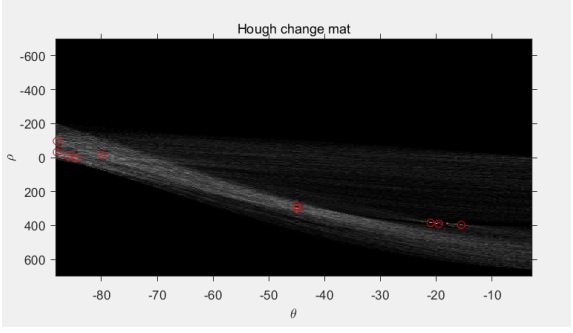
第一步：车道线的图片预处理，使应用霍夫变换的对象为边缘图像
设置阈值和生成边缘图像：



第二步：应用霍夫变换函数，获取图片中的直线

应用霍夫变换时应注意，我们的目标是识别车道线，但是图片的边框会对直线的判别造成干扰，图片中的地平线、电线杆、建筑物等物体也会对车道线是识别造成干扰，**我们可以发现，边框、地平线、电线杆、建筑物的轮廓是水平和竖直的，那么我们在确定霍夫变换θ的范围时，可以舍弃掉水平线和竖直的线。**由经验可知，两边的车道线摆动范围为 40 度~60 度，这个条件可以帮助我们去掉一些干扰，只对该范围内的直线感兴趣。从(x,y)切换到(ρ, θ)后，对(ρ, θ)平面进行**投票统计**，某个点的票数越高，则对应到(x,y)平面，在该直线上的点就越多，这些点可构成一条直线。我们取(ρ, θ)平面的 10 个**峰值**，也就是取出(x,y)平面的 10 条直线进行选择。
霍夫变换域图像，其中红色圈部分为 10 个峰值点。图为左边和右边的霍夫变换域





第三步：筛选获得的直线

霍夫变换和取 10 个峰值点完成后，转到(x,y)平面进行筛选，得到每条直线的两个端点，求出两个端点的欧几里得距离，选取最长的一个为识别的最终结果。

第四步：画出车道线。

因为左边与右边都要取最长的一条，如果合在一起求最长线段的话只能得到一条，若选择第一长的和第二长的线段，则这两条线段不一定就是左右两边的车道线。所以分开左右求解，得到结果后再整合。

原图：

识别结果：**左边**

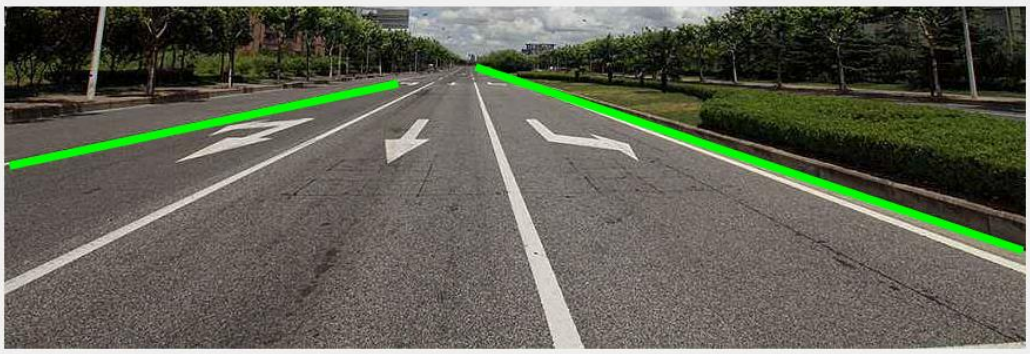
识别结果：**右边**

识别结果：**整合**

识别结果 2

原图：

霍夫变换应用后：



识别结果 3

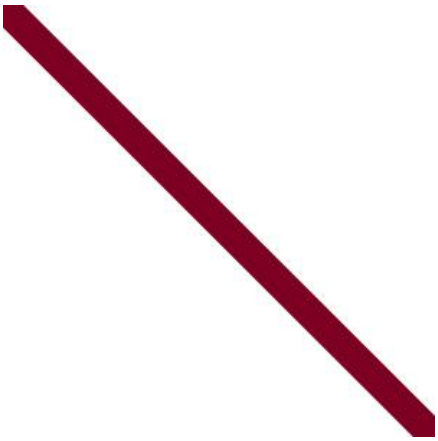
原图：

应用霍夫变换识别后：

误差分析

为了观察霍夫变换的识别效果，对我们编写的算法进行量化测试，给定一个固定斜率的直线，对该直线进行霍夫变换，得到识别后的斜率，与固定斜率进行误差分析，检验算法效果。

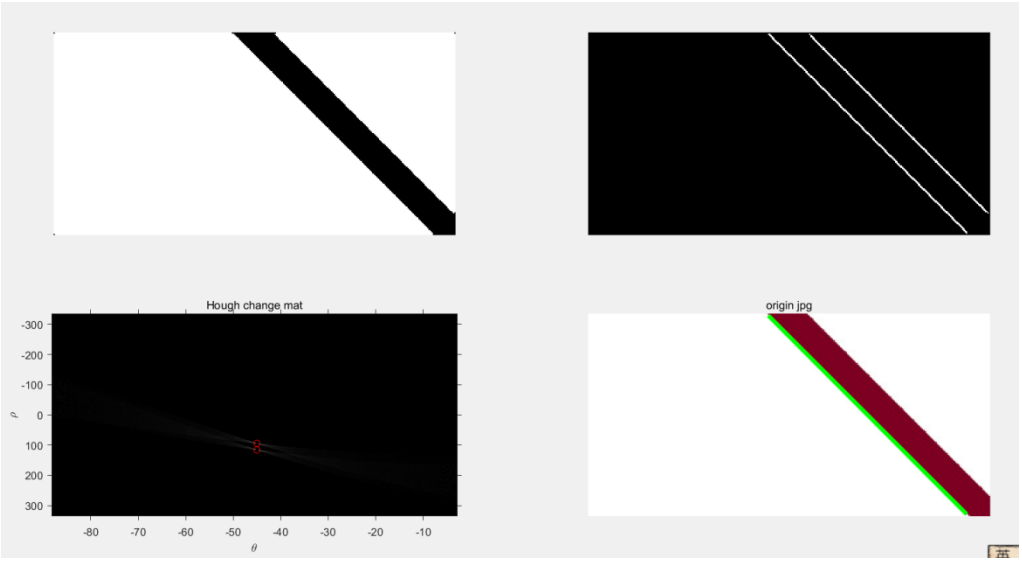
设定固定斜率 k=-1 的图片，进行误差分析



原图

我们的算法识别过程：

选取阈值→取边缘→应用霍夫变换→识别直线（绿色的线为我们算法识别的直线）



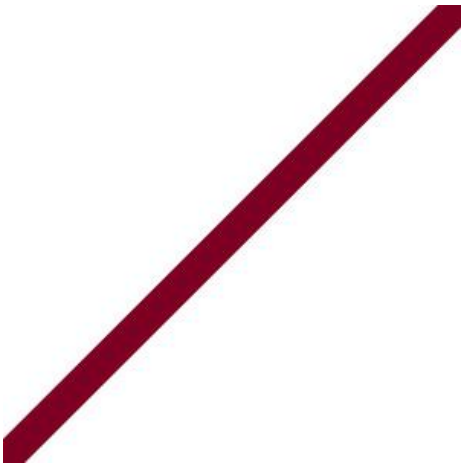
我们识别的直线斜率为（matlab 命令行窗口）

```
>> k

k =

   -1.112781954887218
```

绝对误差和相对误差为：

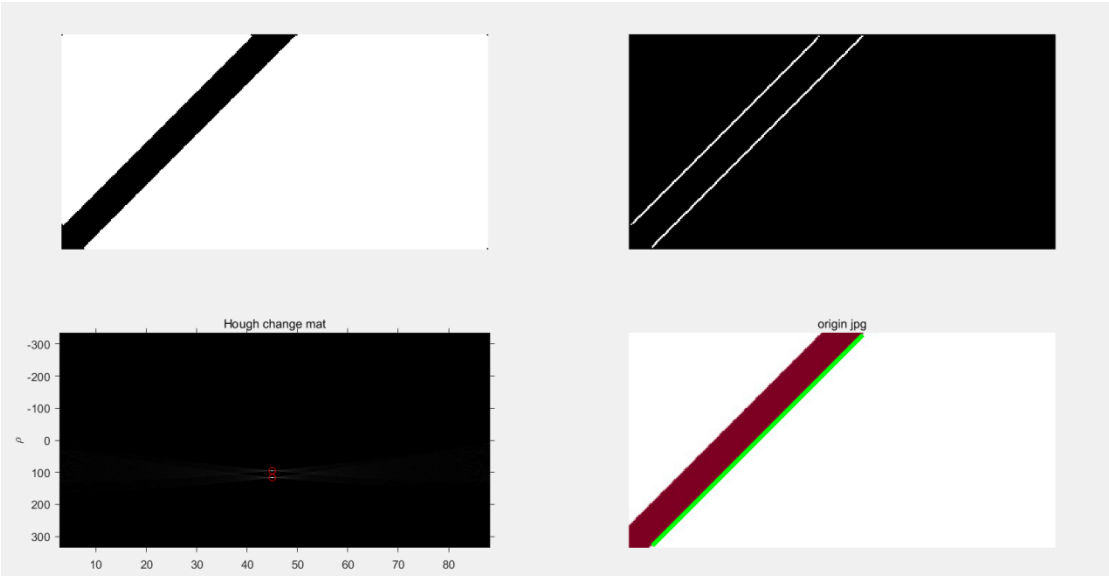
```
Ex =

    0.112781954887218


Rx =

    0.112781954887218
```

设定固定斜率为 1 的图片，进行误差分析



原图

选取阈值→取边缘→应用霍夫变换→识别直线（绿色的线为我们算法识别的直线）



斜率： （matlab 命令行窗口）

```
k =

    1.112781954887218
```

绝对误差与相对误差：

```
Ex =

    0.112781954887218


Rx =

    0.112781954887218
```

曲线拟合应用

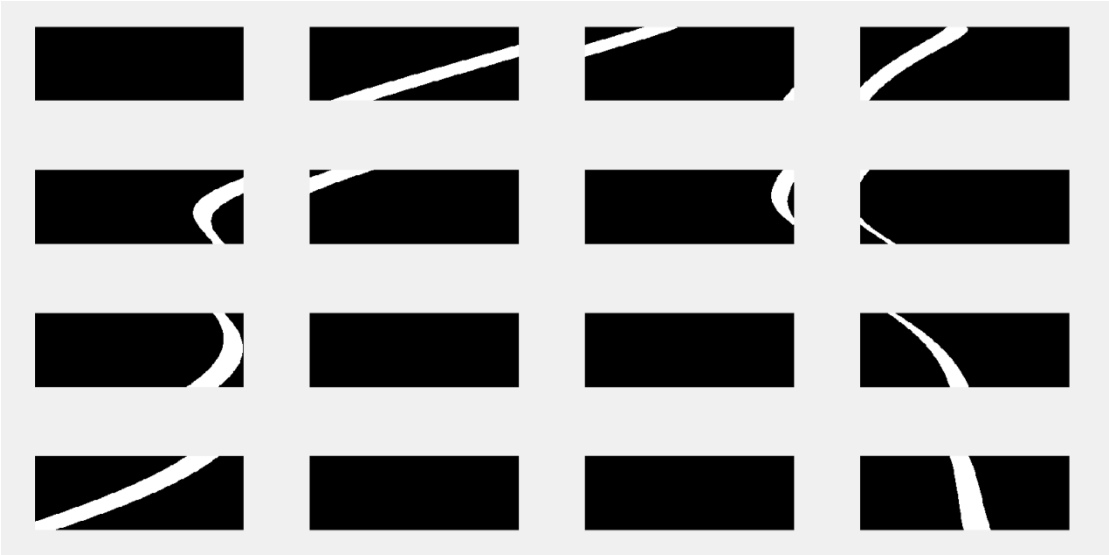
在现实生活中，车道线并不是完全是直线，可能是弯道，所以车道线识别不能仅仅局限于识别直线，弯道也需要识别。

在探索识别弯道的过程中，我们首先想到的是提取边缘曲线的解析式，然后通过解析式确定车道的轨迹。但是边缘函数曲线的提取很困难，并且还会受到很多不相干的边缘信息的干扰，比如路边绿化树和绿化带的影响。

于是我们另辟蹊径，应用《数值计算方法》中课堂上所学到的知识来识别弯道。

原图：

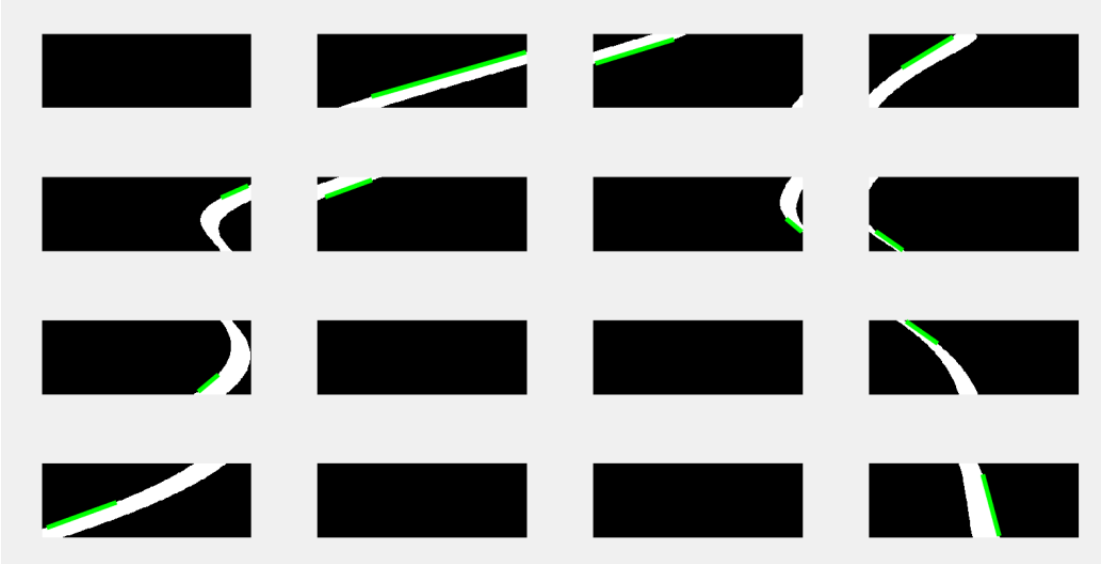
第一步：对图像预处理后，对感兴趣的 ROI 区域进行分割称为 4*4 的子图



第二步：

分别对每个子图应用霍夫变换，这里运用了直线逼近曲线的方法，当子图足够小时，曲线可用直线代替。为了看到分割效果，选取 4*4 的分割，可以分割成更小。

（**绿色的线**为霍夫变换应用在子图上所得到的结果）

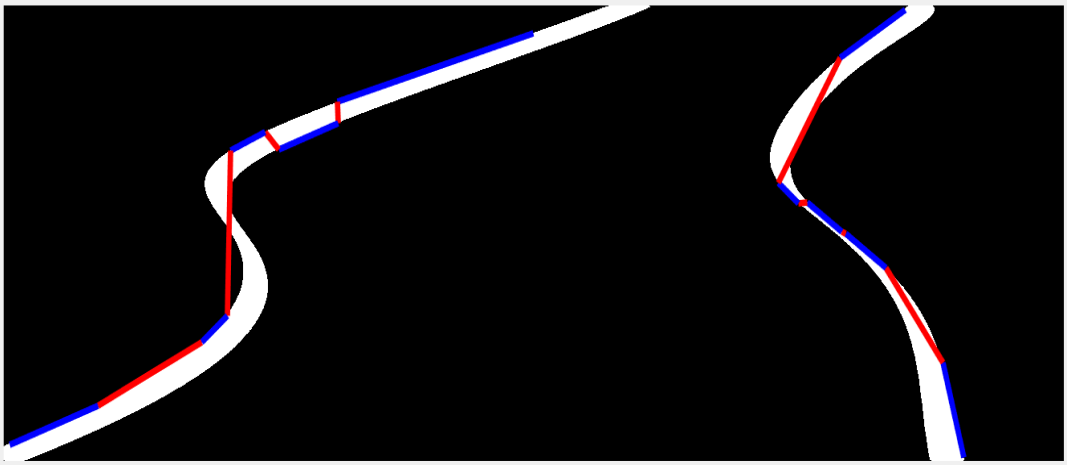


第三步：每个子图应用霍夫变换后，保留每个子图中最长线段的两个端点。

这里值得注意的是，子图中全黑的部分没有检测到实现，所以预设两个端点的坐标都是（0,0）并且每个子图都带有一个布尔变量，用于标记该子图中是否有直线被检测到，这样便于用折线来逼近曲线的过程中点与点之间的连接问题的判断。

把每一个子图的端点，与它周围的 8 幅子图进行相关计算，哪两个端点间欧几里得距离最近，则把这两个端点相连。

如图所示，蓝色线段为子图内部最长线段，红色部分为子图与子图之间的连线

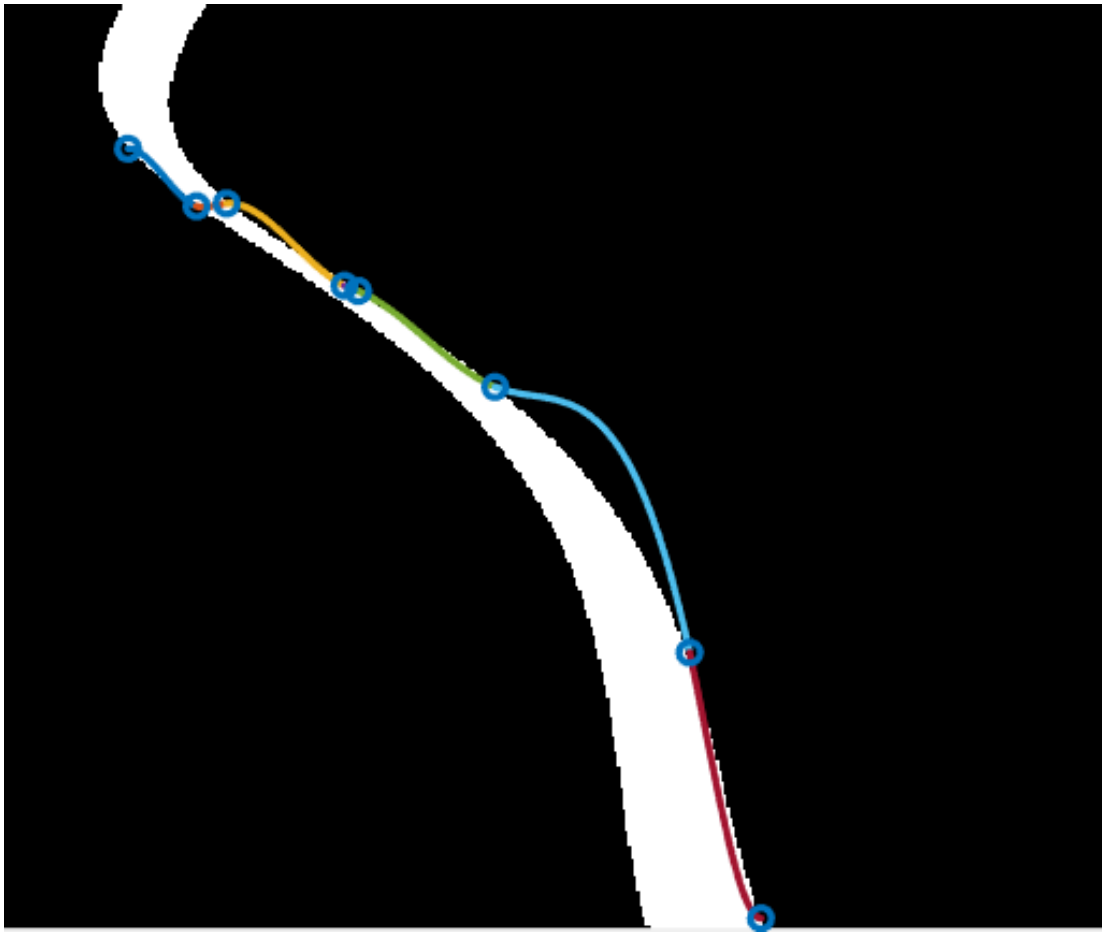


连接完成后，我们实现了用折线逼近曲线的任务，虽然曲线的细节没有很好的体现，但是曲线总体的走向趋势的信息被保留，这个方案是可行性高的。

第三步的改进：获取每一幅子图中线段端点的信息，用这些点的坐标进行拟合。

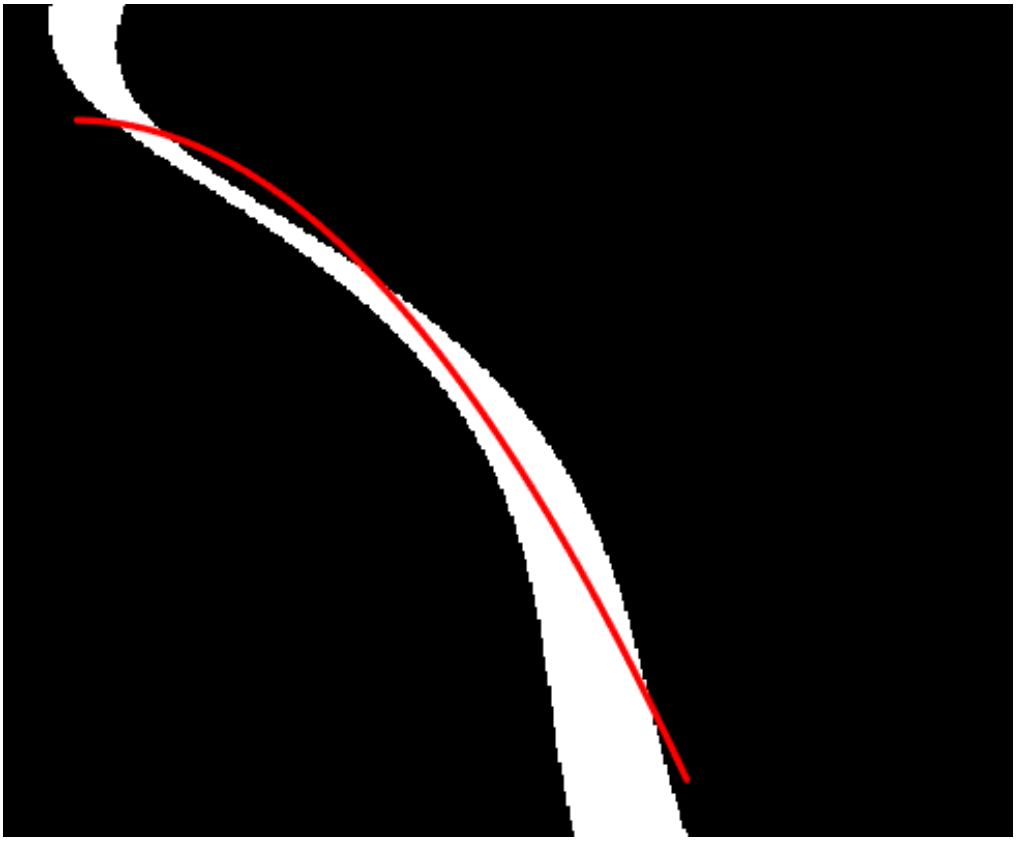
我们使用的方法是三次紧压样条函数来进行差值运算。运行结果为如下图：

三次紧压样条函数差值拟合



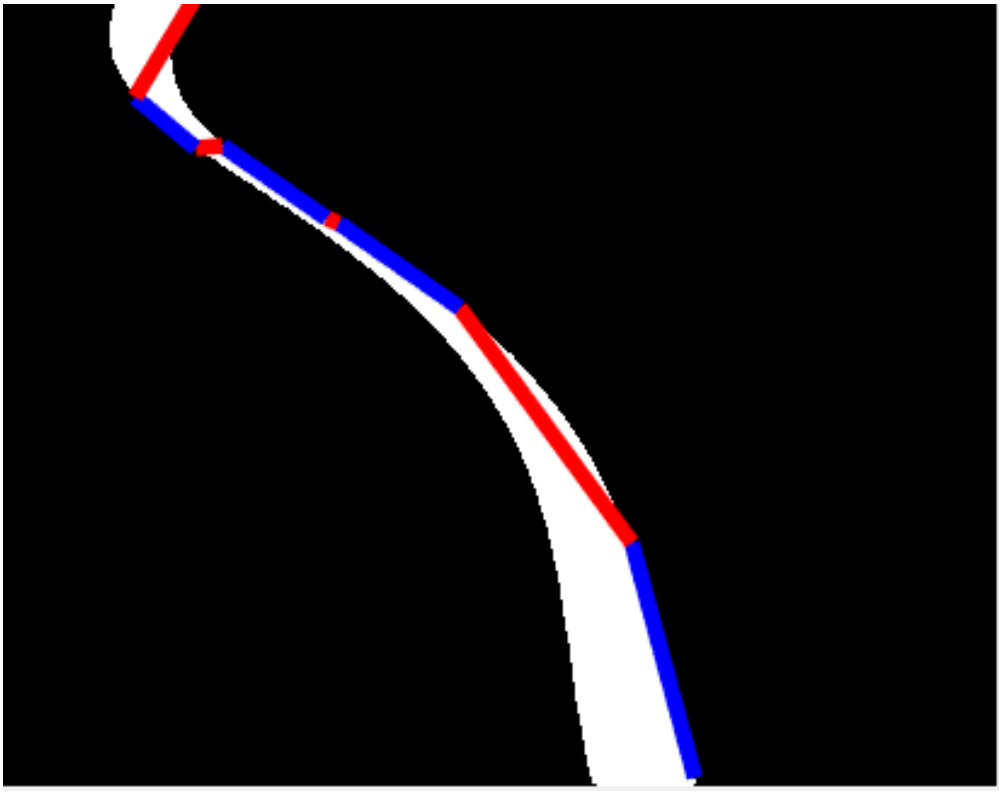
可以看到三次紧压样条函数差值可以获取曲线弯曲走向的细节，凹凸可以表现出来

最小二乘法进行拟合



可以看到最小二乘法能够描述出曲线的大致走向，并且比折线的情况更加平滑

简单的折线表示



可以看出用三次紧压样条函数和最小二乘法来进行差值运算，能够获取部分曲线细节的信息，优于折线所显示的信息，但是在实际应用中，我们可以先用折线来获取曲线的趋势走向，如果需要曲线的细节，再进行曲线的拟合。

结论：

从以上的算法分析和实验结果，可以看出本论文中的算法能对车道线较好较准确的进行识别，算法主要特点和优点有：

1．使用 Canny 算子进行边缘检测，能够在噪声比较强的情况下识别出令人比较满意的边缘情况，或者说保留了大部分边缘信息。

2．在进行霍夫变换之前进行了简单而有效的预处理，使得在不大幅增加算法时间复杂度的基础上尽可能提高了识别效果；

3．对于曲线的处理比较特殊，考虑到一般的曲线拟合虽然拟合效果很好，但是效率比较低或者说时间复杂度大，在实际应用中特别是实时处理会对计算机有较高要求，因此我们采用将图像分割的方法，使得在每个分块内近似为直线，然后对于每个分块使用霍夫算法，最后将这些直线连接起来，用折线逼近曲线。这样使得我们在尽可能小地影响识别效果的前提下，提高算法效率。

在实际操作中我们也发现了一些问题，主要有：

1．对于复杂路况（曲率较大的弯道）和天气条件欠佳时，识别效果比较差；

算法的自适应性还不够，特别是对于各种滤波的参数并没有做到自适应图像变换，所以对于某些特定的场景识别效果也不尽人意。

下一步我们打算从以下几个方面继续改进我们的算法，一方面是继续优化预处理，更好的移除噪声干扰，另外一方面是参考各文献，优化用于直线识别的 Hough 算法，还是有比较多的工作去做。

2．对于未来，随着人工智能、机器学习等的不断发展，将这些技术应用于车道识别中，必将能极大的提高车道线识别的精确性和适应性，并且伴随着智能汽车的进一步发展，新的技术不断得到应用，车道线的识别技术一定会迎来一个大的发展。特别是国内，随着中国的科技水平不断提高，汽车行业快速崛起，市场需要持续增大，对于车道线的自动识别的需求也将更加迫切。所以未来是光明的，但更需要我们脚踏实地，不断取得新的突破。

参考文献

(1),<Numerical Methods Using MATLAB,Fourth Edition>,(Mathews,J.H) 电子工业出版社
(2)《MATLAB R2015a 数字图像处理》（丁伟雄编著）清华大学出版社
(3)《数字图像中边缘检测算法研究》（刘仁云，孙秋成，王春艳著）科学出版社
(4)《数字图像理解与智能技术----基于 MATLAB 和 VC++实现》（孙明主编）电子工业出版社
（5）鲁曼,蔡自兴,李仪.道路区域分割的车道线检测方法[J].智能系统学报,2010,5(6):505-509
（6）杨喜宁,段建民,高德芝,郑榜贵.基于改进 Hough 变换的车道线检测技术[J].计算机测量与控制,2010,18(2):292-298
（7）Robust Lane Detection Based On Convolutional Neural Network and Random Sample Consensus, Jihun Kim, Minho Lee, 2014
（8）Robust Lane Detection using Two-stage Feature Extraction with Curve Fitting, Jianwei Niu, Jie Lu, Mingliang Xu, Pei Lv, Xiaoke Zhao,2015

附录：代码

1. Csfit 三次紧压样条函数

```
function S=csfit(X,Y,dx0,dxn)
    N=length(X)-1;
    H=diff(X);
    D=diff(Y)./H;
    A=H(2:N-1);
    B=2*(H(1:N-1)+H(2:N));
    C=H(2:N);
    U=6*diff(D);
    B(1)=B(1)-H(1)/2;
    U(1)=U(1)-3*(D(1)-dx0);
    B(N-1)=B(N-1)-H(N)/2;
    U(N-1)=U(N-1)-3*(dxn-D(N));

    for k=2:N-1
        temp=A(k-1)/B(k-1);
        B(k)=B(k)-temp*C(k-1);
        U(k)=U(k)-temp*U(k-1);
    end

    M(N)=U(N-1)/B(N-1);
```

```
    for k=N-2:-1:1
        M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
    end

    M(1)=3*(D(1)-dx0)/H(1)-M(2)/2;
    M(N+1)=3*(dxn-D(N))/H(N)-M(N)/2;

    for k=0:N-1
        S(k+1,1)=(M(k+2)-M(k+1))/(6*H(k+1));
        S(k+1,2)=M(k+1)/2;
        S(k+1,3)=D(k+1)-H(k+1)*(2*M(k+1)+M(k+2))/6;
        S(k+1,4)=Y(k+1);
    end
end
```

Ispoly.m

```
function C=lspoly(X,Y,M)

n=length(X);

B=zeros(1:M+1);

F=zeros(n,M+1);

for k=1:M+1

    F( : ,k)=X'.^(k-1);

end

A=F'*F;

B=F'*Y';

C=A\B;

C=flipud(C);
```

2. curve_fitting.m 获取坐标信息

```
function s=curve_fitting(process3)
% 函数功能 获得端点信息并且通过坐标的方式返回
% 分割图像
[width,height,d11,d12,d13,d14,d21,d22,d23,d24,d31,d32,d33,d34,d
41,d42,d43,d44]=division(process3);
%获取坐标信息
[B11,long11]=divimg(d11);
[B12,long12]=divimg(d12);
[B13,long13]=divimg(d13);
[B14,long14]=divimg(d14);

[B21,long21]=divimg(d21);
[B22,long22]=divimg(d22);
[B23,long23]=divimg(d23);
[B24,long24]=divimg(d24);

[B31,long31]=divimg(d31);
[B32,long32]=divimg(d32);
[B33,long33]=divimg(d33);
[B34,long34]=divimg(d34);

[B41,long41]=divimg(d41);
```

```
[B42,long42]=divimg(d42);
[B43,long43]=divimg(d43);
[B44,long44]=divimg(d44);
x=[0,0,0,0,0,0,0,0];
y=[0,0,0,0,0,0,0,0];

% 左边
% if(B12~=0)
%    long12(1,1)=long12(1,1)+width;
%    long12(2,1)=long12(2,1)+width;
%       x(1)=long12(1,1);
%       x(2)=long12(2,1);
%       y(1)=long12(1,2);
%       y(2)=long12(2,2);
% end
%
% if(B22~=0)
%    long22(1,1)=long22(1,1)+width;
%    long22(2,1)=long22(2,1)+width;
%    long22(1,2)=long22(1,2)+height;
%    long22(2,2)=long22(2,2)+height;
%       x(3)=long22(1,1);
%       x(4)=long22(2,1);
```

```
%       y(3)=long22(1,2);
%       y(4)=long22(2,2);
% end
%
% if(B21~=0)
%    long21(1,2)=long21(1,2)+height;
%    long21(2,2)=long21(2,2)+height;
%       x(5)=long21(1,1);
%       x(6)=long21(2,1);
%       y(5)=long21(1,2);
%       y(6)=long21(2,2);
% end
%
% if(B31~=0)
%    long31(1,2)=long31(1,2)+2*height;
%    long31(2,2)=long31(2,2)+2*height;
%       x(7)=long31(1,1);
%       x(8)=long31(2,1);
%       y(7)=long31(1,2);
%       y(8)=long31(2,2);
% end
%
```

```
% if(B41~=0)
%    long41(1,2)=long41(1,2)+3*height;
%    long41(2,2)=long41(2,2)+3*height;
%       x(9)=long41(1,1);
%       x(10)=long41(2,1);
%       y(9)=long41(1,2);
%       y(10)=long41(2,2);
% end
% if(B14~=0)
%    long14(1,1)=long14(1,1)+3*width;
%    long14(2,1)=long14(2,1)+3*width;
%       x(1)=long14(1,1);
%       x(2)=long14(2,1);
%       y(1)=long14(1,2);
%       y(2)=long14(2,2);
% end

% 右边
if(B23~=0)
     long23(1,1)=long23(1,1)+2*width;
     long23(2,1)=long23(2,1)+2*width;
     long23(1,2)=long23(1,2)+height;
```

```
        long23(2,2)=long23(2,2)+height;
        x(1)=long23(1,1);
        x(2)=long23(2,1);
        y(1)=long23(1,2);
        y(2)=long23(2,2);
    end

    if(B24~=0)
        long24(1,1)=long24(1,1)+3*width;
        long24(2,1)=long24(2,1)+3*width;
        long24(1,2)=long24(1,2)+height;
        long24(2,2)=long24(2,2)+height;
        x(3)=long24(1,1);
        x(4)=long24(2,1);
        y(3)=long24(1,2);
        y(4)=long24(2,2);
    end

    if(B34~=0)
        long34(1,1)=long34(1,1)+3*width;
        long34(2,1)=long34(2,1)+3*width;
        long34(1,2)=long34(1,2)+2*height;
        long34(2,2)=long34(2,2)+2*height;
```

```
        x(5)=long34(1,1);
        x(6)=long34(2,1);
        y(5)=long34(1,2);
        y(6)=long34(2,2);
    end

    if(B44~=0)
        long44(1,1)=long44(1,1)+3*width;
        long44(2,1)=long44(2,1)+3*width;
        long44(1,2)=long44(1,2)+3*height;
        long44(2,2)=long44(2,2)+3*height;
        x(7)=long44(1,1);
        x(8)=long44(2,1);
        y(7)=long44(1,2);
        y(8)=long44(2,2);
    end
    %定义结构体
    %冒泡排序
    s=struct('x',x,'y',y);
    for i=1:8
        for j=i+1:8
            if(s.x(i)>s.x(j))
                xx=s.x(i);
```

```
                yy=s.y(i);
                s.x(i)=s.x(j);
                s.y(i)=s.y(j);
                s.x(j)=xx;
                s.y(j)=yy;
            end
        end
    end
end
```

3. divimg 检测每个子图中的最长线段

```
function [B,longline]=divimg(D)

    % 函数功能 检测每个子图中的最长线段 并返回端点和标记
    BW=edge(D,'canny');% 求图像边缘 用canny算子
    theta1=3:0.5:88;
    theta2=-88:0.5:-3;
    n=1.*(theta1)+1.*(theta2); % 选取theta范围
    [H,T,R]=hough(BW,'RhoResolution',0.5,'Theta',n);
    P=houghpeaks(H,10,'threshold',ceil(0.5*max(H(:))));% 用于在Hough变换后矩阵中寻找最佳 H
    lines=houghlines(BW,T,R,P,'FillGap',5,'MinLength',3);%根据霍夫变换提取线段
```

```
    max_len=0;% 选择子图中最长的线段
    xy_long=[0,0;0,0];
    for k=1:length(lines)
        xy=[lines(k).point1;lines(k).point2];
        len=norm(lines(k).point1 - lines(k).point2);
        if(len>=0)
            if(len > max_len)
                max_len=len;
                xy_long=xy;
            end
        end
    end
    % 如果没有检测到线段则设两个端点为（0,0） B=0 否则 B=1，返回直线端点
    if(xy_long(1,1)==0 && xy_long(1,2)==0 && xy_long(2,1)==0 && xy_long(2,2)==0)
        B=0;
        longline=[0,0;0,0];
    else
        B=1;
        longline=xy_long;
    end
```

```
end
```

4. division.m 分割图像

```
function[width,height,d11,d12,d13,d14,d21,d22,d23,d24,d31,d32,d
33,d34,d41,d42,d43,d44]=division(process3)
% 函数功能 分割图像 分成4*4=16个子图
% d11~d44为分割的子图
[n,m]=size(process3);
% rect =[左上角点纵坐标(xmin), 左上角点横坐标(ymin),
col(xWidth), row(yHeight)]
%为了图像的长宽为4的倍数
rn=mod(n,4);
rm=mod(m,4);
n=n-rn;
m=m-rm;
height=n/4;
width=m/4;
% 子图第一行
d11=imcrop(process3,[0,0,width,height]);
d12=imcrop(process3,[1*width,0,width,height]);
d13=imcrop(process3,[2*width,0,width,height]);
d14=imcrop(process3,[3*width,0,width,height]);

% 子图第二行
```

```
d21=imcrop(process3,[0,height,width,height]);
d22=imcrop(process3,[1*width,height,width,height]);
d23=imcrop(process3,[2*width,height,width,height]);
d24=imcrop(process3,[3*width,height,width,height]);

% 子图第三行
d31=imcrop(process3,[0,2*height,width,height]);
d32=imcrop(process3,[1*width,2*height,width,height]);
d33=imcrop(process3,[2*width,2*height,width,height]);
d34=imcrop(process3,[3*width,2*height,width,height]);

% 子图第四行
d41=imcrop(process3,[0,3*height,width,height]);
d42=imcrop(process3,[1*width,3*height,width,height]);
d43=imcrop(process3,[2*width,3*height,width,height]);
d44=imcrop(process3,[3*width,3*height,width,height]);

% 画出子图 这部分用于检测结果 所以注释
% subplot(4,4,1);imshow(d11);
% subplot(4,4,2);imshow(d12);
% subplot(4,4,3);imshow(d13);
% subplot(4,4,4);imshow(d14);
%
% subplot(4,4,5);imshow(d21);
```

```
% subplot(4,4,6);imshow(d22);
% subplot(4,4,7);imshow(d23);
% subplot(4,4,8);imshow(d24);
%
% subplot(4,4,9);imshow(d31);
% subplot(4,4,10);imshow(d32);
% subplot(4,4,11);imshow(d33);
% subplot(4,4,12);imshow(d34);
%
% subplot(4,4,13);imshow(d41);
% subplot(4,4,14);imshow(d42);
% subplot(4,4,15);imshow(d43);
% subplot(4,4,16);imshow(d44);
end
```

5. Divisiondetect.m画出每个子图中检测到的线段

```
RGB=imread('jiangdi.jpg');% 读入图像
[process3,processcrop]=preprocess(RGB); % 预处理图像
% 分割图像
[width,height,d11,d12,d13,d14,d21,d22,d23,d24,d31,d32,d33,d34,d
41,d42,d43,d44]=division(process3);
% 画出每个子图中最长的线段
subplot(4,4,1);imshow(d11);hold on;
```

```
[B11,long11]=divimg(d11);
plot(long11(:,1),long11(:,2),'LineWidth',4,'Color','green');

subplot(4,4,2);imshow(d12);hold on;
[B12,long12]=divimg(d12);
plot(long12(:,1),long12(:,2),'LineWidth',4,'Color','green');

subplot(4,4,3);imshow(d13);hold on;
[B13,long13]=divimg(d13);
plot(long13(:,1),long13(:,2),'LineWidth',4,'Color','green');

subplot(4,4,4);imshow(d14);hold on;
[B14,long14]=divimg(d14);
plot(long14(:,1),long14(:,2),'LineWidth',4,'Color','green');

subplot(4,4,5);imshow(d21);hold on;
[B21,long21]=divimg(d21);
plot(long21(:,1),long21(:,2),'LineWidth',4,'Color','green');

subplot(4,4,6);imshow(d22);hold on;
[B22,long22]=divimg(d22);
plot(long22(:,1),long22(:,2),'LineWidth',4,'Color','green');

subplot(4,4,7);imshow(d23);hold on;
[B23,long23]=divimg(d23);
```

```
plot(long23(:,1),long23(:,2),'LineWidth',4,'Color','green');

subplot(4,4,8);imshow(d24);hold on;
[B24,long24]=divimg(d24);
plot(long24(:,1),long24(:,2),'LineWidth',4,'Color','green');

subplot(4,4,9);imshow(d31);hold on;
[B31,long31]=divimg(d31);
plot(long31(:,1),long31(:,2),'LineWidth',4,'Color','green');

subplot(4,4,10);imshow(d32);hold on;
[B32,long32]=divimg(d32);
plot(long32(:,1),long32(:,2),'LineWidth',4,'Color','green');

subplot(4,4,11);imshow(d33);hold on;
[B33,long33]=divimg(d33);
plot(long33(:,1),long33(:,2),'LineWidth',4,'Color','green');

subplot(4,4,12);imshow(d34);hold on;
[B34,long34]=divimg(d34);
plot(long34(:,1),long34(:,2),'LineWidth',4,'Color','green');

subplot(4,4,13);imshow(d41);hold on;
[B41,long41]=divimg(d41);
plot(long41(:,1),long41(:,2),'LineWidth',4,'Color','green');
```

```
subplot(4,4,14);imshow(d42);hold on;
[B42,long42]=divimg(d42);
plot(long42(:,1),long42(:,2),'LineWidth',4,'Color','green');

subplot(4,4,15);imshow(d43);hold on;
[B43,long43]=divimg(d43);
plot(long43(:,1),long43(:,2),'LineWidth',4,'Color','green');

subplot(4,4,16);imshow(d44);hold on;
[B44,long44]=divimg(d44);
plot(long44(:,1),long44(:,2),'LineWidth',4,'Color','green');
```

6.leftandright 检测车道线

```
RGB=imread('baiyun.jpg');% 读取图像
[process3,processcrop]=preprocess(RGB);% 图像预处理
imshow(processcrop);hold on;
xy_long1=longestlineleft(process3);% 画出左边的车道线
plot(xy_long1(:,1),xy_long1(:,2),'LineWidth',4,'Color','green')
xy_long2=longestlineright(process3);%画出右边的车道线
plot(xy_long2(:,1),xy_long2(:,2),'LineWidth',4,'Color','green')
```

7. longestlineleft.m 检测左边的车道线

```
function xy_long=longestlineleft(process3)
BW=edge(process3,'canny'); % canny算子边缘检测
```

```
theta1=3:0.5:88;
[H,T,R]=hough(BW,'RhoResolution',0.5,'Theta',theta1);% 应用霍夫变换
P=houghpeaks(H,10,'threshold',ceil(0.3*max(H(:))));
% houghpeaks用于在Hough变换后矩阵中寻找最佳 H
% 10是numpeaks 峰值数目 提取hough变换后参数平面的峰值点 返回P（峰值点）乘2的矩阵 包含行坐标和列坐标
% 'threshold'和ceil(0.3*max(H(:)))指定寻找峰值的门限和对峰值周围图像点的抑制范围

lines=houghlines(BW,T,R,P,'FillGap',10,'MinLength',7);%根据霍夫变换提取线段
% houghlines(BW,theta,rho,peaks) BW边缘 lines的长度是提取线段数目 是结构矩阵
% 'FillGap',5,'MinLength',7 用于指定是否合并或保留线段

max_len=0;
for k=1:length(lines)
    xy=[lines(k).point1;lines(k).point2];
    len=norm(lines(k).point1 - lines(k).point2);% 两个端点的欧几里得距离
    if(len>=30)
        if(len > max_len)
```

```
                max_len=len;
                xy_long=xy;
            end
        end
    end
    %
plot(xy_long(:,1),xy_long(:,2),'LineWidth',4,'Color','green');
% 画出最长的线段
    % k=abs((xy_long(2,2)-xy_long(1,2))/(xy_long(1,2)-
xy_long(1,1))); %斜率
    % Ex=abs(1-k) %绝对误差
    % Rx=Ex/abs(1) %相对误差
end
% 函数功能 检测左边车道线
```

8. longestlineright.m 检测右边的车道线

```
function xy_long=longestlineright(process3)
BW=edge(process3,'canny'); % canny算子边缘检测
theta2=-88:0.5:-3;
[H,T,R]=hough(BW,'RhoResolution',0.5,'Theta',theta2);% 应用霍夫
变换
P=houghpeaks(H,10,'threshold',ceil(0.3*max(H(:))));
```

```
% houghpeaks用于在Hough变换后矩阵中寻找最佳 H
% 10是numpeaks 峰值数目 提取hough变换后参数平面的峰值点 返回P（峰值点）乘2的矩阵 包含行坐标和列坐标
% 'threshold'和ceil(0.3*max(H(:)))指定寻找峰值的门限和对峰值周围图像点的抑制范围

lines=houghlines(BW,T,R,P,'FillGap',10,'MinLength',7);%根据霍夫变换提取线段
% houghlines(BW,theta,rho,peaks) BW边缘 lines的长度是提取线段数目 是结构矩阵
% 'FillGap',5,'MinLength',7 用于指定是否合并或保留线段

max_len=0;
for k=1:length(lines)
    xy=[lines(k).point1;lines(k).point2];
    len=norm(lines(k).point1 - lines(k).point2);% 两个端点的欧几里得距离
    if(len>=30)
        if(len > max_len)
            max_len=len;
            xy_long=xy;
        end
    end
```

```
end
%
plot(xy_long(:,1),xy_long(:,2),'LineWidth',4,'Color','green');
% 画出最长的线段
% k=abs((xy_long(2,2)-xy_long(1,2))/(xy_long(1,2)-
xy_long(1,1))); %斜率
% Ex=abs(1-k) %绝对误差
% Rx=Ex/abs(1) %相对误差
end
% 函数功能 检测右边车道线
```

9. test1 和 test2 使用最小二乘法和三次紧缩样条函数

test1 最小二乘法测试

```
RGB=imread('jiangdi.jpg');
[process3,processcrop]=preprocess(RGB);% 预处理
s=curve_fitting(process3);% 三次紧压样条函数
imshow(process3);hold on;
X=s.x;
Y=s.y;
C=lspoly(X,Y,2);
x1=X(1):.01:X(8);y1=C(1,1)*x1.^2+C(2,1).*x1+C(3,1);
plot(x1,y1,'.','LineWidth',4,'Color','red');
```

test2 三次紧缩样条函数测试

```
% 测试三次紧压样条函数

RGB=imread('jiangdi.jpg');

[process3,processcrop]=preprocess(RGB);% 预处理

s=curve_fitting(process3);% 三次紧压样条函数

[n,m]=size(process3);

imshow(process3);hold on;

X=s.x;

Y=s.y;

S=csfit(X,Y,0,0);

x1=X(1):0.01:X(2); y1=polyval(S(1,:),x1-X(1));

x2=X(2):0.01:X(3); y2=polyval(S(2,:),x2-X(2));

x3=X(3):0.01:X(4); y3=polyval(S(3,:),x3-X(3));

x4=X(4):0.01:X(5); y4=polyval(S(4,:),x4-X(4));

x5=X(5):0.01:X(6); y5=polyval(S(5,:),x5-X(5));

x6=X(6):0.01:X(7); y6=polyval(S(6,:),x6-X(6));

x7=X(7):0.01:X(8); y7=polyval(S(7,:),x7-X(7));


plot(x1,y1,'.','LineWidth',4);

plot(x2,y2,'.','LineWidth',4);

plot(x3,y3,'.','LineWidth',4);

plot(x4,y4,'.','LineWidth',4);
```

```
plot(x5,y5,'.','LineWidth',4);
plot(x6,y6,'.','LineWidth',4);
plot(x7,y7,'.','LineWidth',4);
plot(X,Y,'o','LineWidth',2);
```

10. preprocess.m 图像预处理

```
function [preprocess3,preprocess1]=preprocess(RGB)
% 函数功能 预处理图像 选取感兴趣区域 滤波去噪声 分割图像
    [n,m]=size(RGB);
    preprocess1=imcrop(RGB,[0 0.5*n m 0.5*n]);
    % xmin ymin width height 预处理1 剪裁图像
    % 选取图像的感兴趣ROI区域 图像的下半部分
    f=fspecial('gaussian',[3 3],10); %高斯滤波 去噪声点效果好
    gauss_smooth = imfilter(preprocess1,f,'same'); % 滤波
    preprocess3=im2bw(gauss_smooth,150/255);
    % 双峰法分割图像 150为阈值
end
```

组员合照

周禅城（左一） 钟诗婷（中间） 钟思根（右一）