**车道线检测**

**1、实践内容与要求**

针对路面标线检测是实现车道偏离预警的关键技术，对于实现车辆的完全自主驾驶具有重要的意义。基于OpenCV库设计程序实现视频中公路车道线的检测，要求能够检测出公路上的车道线并用指定颜色标示，结果在显示器显示，同时存储为一个新的视频文件。

**2、设计思路**

**2.1 图片灰度化、二值化处理**

通过灰度化处理，将图像的三通道转化为单通道，便于后续数据处理。考虑到后续需要进行图像的平滑处理来降低噪声干扰，故还需要对图像进行二值化处理。

**2.2 平滑图像处理**

为减小图像中其他环境的影响，对图片进行腐蚀、膨胀等操作完成降噪。

**2.3 Canny边缘检测**

为了突出车道线，需要对处理后的图像做边缘处理。由于车道线为白色或者黄色，地面通常为黑色或者灰色，因此车道线边缘处会有明显的明暗交替。

常见的边缘检测算法有Canny边缘检测算法和Sobel边缘检测算法，由于Canny边缘检测算法包含了更多步骤，其精度和鲁棒性更高，故在该项目中采取Canny边缘检测算法。

Canny边缘检测算法通常包含以下步骤：降噪处理、灰度转换、计算每个点的梯度大小和方向、使用非极大值抑制消除边缘检测带来的杂散效应、应用双阈值确定边缘、通过抑制弱边缘。

在该程序中可直接应用Canny()函数即可实现边缘检测。

**2.4 ROI区域提取**

为了实现车道线检测，需要将ROI区域提取出来。为实现该截取功能，可以在程序中定义一个region\_of\_interest函数，传入经过平滑处理的图像、梯形的四个角点坐标，该函数返回ROI区域内经边缘检测提取的结果。

**2.5 霍夫线变换检测直线**

OpenCV提供了霍夫变换检测直线的函数，通过设置不同的参数，检测不同长度的线段。由于车道线存在虚线的可能，因此线段的检测长度不能设置太长，否则短线段会被忽略掉。通过HoughLinesP()函数传入相应参数进行直线的检测。

**2.6 最小二乘法进行直线拟合**

霍夫变换得到的一系列线段结果跟输出结果还是有些差异。为了解决这些差异，需要对我们检测到的数据做一定的后处理操作。

实现以下两步后处理，才能真正得到输出结果。

1、计算左右车道线的直线方程

根据每个线段在图像坐标系下的斜率，判断线段为左车道线还是右车道线，并存于不同的变量中。随后对所有左车道线上的点、所有右车道线上的点做一次最小二乘直线拟合，得到的即为最终的左、右车道线的直线方程。

2、计算左右车道线的上下边界

考虑到现实世界中左右车道线一般都是平行的，所以可以认为左右车道线上最上和最下的点对应的y值，就是左右车道线的边界。

基于以上两步数据后处理的思路，重新定义draw\_lines()函数，将数据后处理过程写入该函数中。根据对线段的后处理，即可得到符合输出要求的两条直线方程的斜率、截距和有效长度。将后处理后的结果绘制在原图上。

**2.7 视频按帧处理，显示车道线**

**3、模块分解**

**3.1 region\_of\_interes函数**

该函数的作用是在输入图像中定义一个梯形区域，并将该区域以外的部分设置为黑色，从而实现对区域内的图像进行提取。函数中的参数包括一个输入图像img和一个包含四个顶点坐标的向量vertices。这四个顶点定义了梯形的形状。

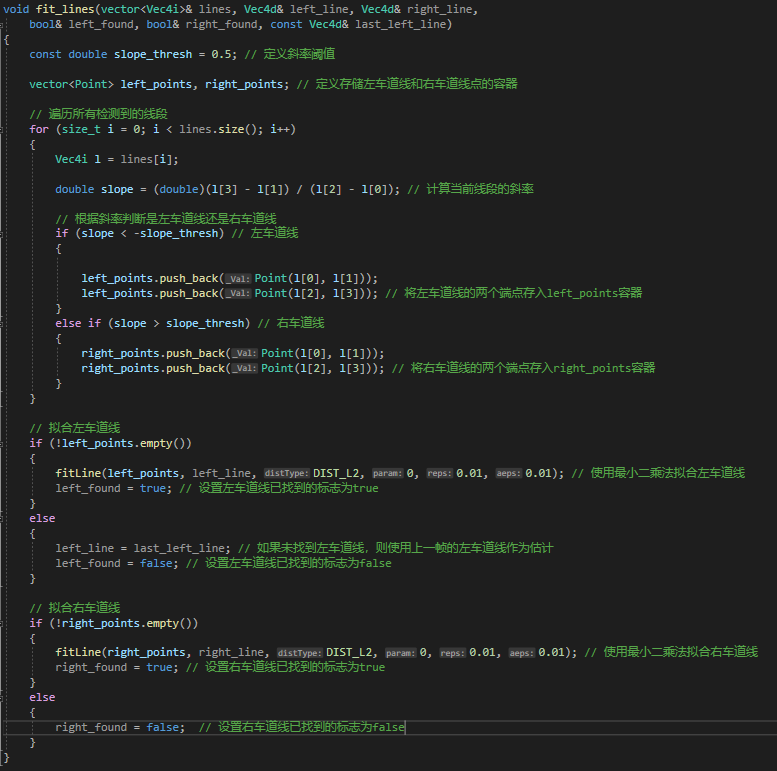
首先，通过调用Mat::zeros函数函数创建一个与输入图像大小相同且类型相同的全黑图像mask，用于存储掩码。然后根据输入图像的通道数判断颜色值ignore\_mask\_color。如果输入图像是多通道的，则将ignore\_mask\_color设置为(255, 255, 255)，表示白色；如果输入图像只有一个通道，则将ignore\_mask\_color设置为255，表示灰度值。接下来，根据传入的顶点坐标，定义了梯形的四个顶点pt1、pt2、pt3、pt4。然后创建一个包含四个顶点的容器pts，表示要绘制的多边形区域。再通过调用fillPoly函数，将pts中定义的多边形填充到mask图像上，颜色为ignore\_mask\_color，从而实现了绘制梯形的操作。最后，通过调用bitwise\_and函数，对输入图像img和掩码图像mask进行按位与操作，将img中不在梯形区域内的部分设置为黑色，得到最终的提取图像masked\_image并返回。

**3.2 fit\_lines函数（核心函数）**

传入参数为线段集合 lines、左车道线和右车道线的输出变量 left\_line 和 right\_line、左车道线和右车道线是否找到的标志变量 left\_found 和 right\_found，以及上一帧的左车道线 last\_left\_line。

定义斜率阈值 slope\_thresh为0.5，该阈值是用于判断线段是左车道线还是右车道线的依据。定义存储左车道线和右车道线点的容器 left\_points 和 right\_points。

接下来使用循环遍历所有检测到的线段并且获取当前线段的起点和终点坐标，并计算线段的斜率。根据斜率判断线段是左车道线还是右车道线。如果斜率小于 -slope\_thresh，则认为是左车道线；如果斜率大于 slope\_thresh，则认为是右车道线。

将左车道线的两个端点存入 left\_points 容器，如果left\_points 容器不为空，即找到了左车道线，然后使用最小二乘法拟合左车道线，并将拟合结果存入 left\_line 变量中。同时将左车道线已找到的标志 left\_found 设置为 true。如果 left\_points 容器为空，即未找到左车道线，则使用上一帧的左车道线 last\_left\_line ，并将它赋值给 left\_line 变量。同时将左车道线已找到的标志 left\_found 设置为 false。右侧车道线同理。

**3.3 intersection\_point函数**

该函数实现了一个求两条直线的交点的函数，使用的是向量叉乘的方法。其中，输入参数line1和line2是由两点表示的直线参数向量，line1[0]和line1[1]表示第一条直线的方向向量，line1[2]和line1[3]表示第一条直线上的一点；同理，line2[0]、line2[1]、line2[2]、line2[3]表示第二条直线上的方向向量和一点。

在函数中，首先根据输入的直线参数向量，计算出两条直线的端点坐标(x1,y1)、(x2,y2)和(x3,y3)、(x4,y4)，然后根据向量叉积公式求出两直线的交点坐标(x,y)。具体过程如下：

1）计算两向量的叉积（即行列式）作为分母：

denominator = (x1 - x2) \* (y3 - y4) - (y1 - y2) \* (x3 - x4);

2）计算分子：

numerator\_x = (x1 \* y2 - y1 \* x2) \* (x3 - x4) - (x1 - x2) \* (x3 \* y4 - y3 \* x4);

numerator\_y = (x1 \* y2 - y1 \* x2) \* (y3 - y4) - (y1 - y2) \* (x3 \* y4 - y3 \* x4);

3）计算交点坐标：

x = numerator\_x / denominator;

y = numerator\_y / denominator;

返回类型为Point类型的交点坐标。

**3.4 main函数**

首先，通过VideoCapture对象打开指定路径下的视频文件，通过VideoWriter创建写入视频的对象。如果无法打开文件，则输出错误信息并返回-1。

接下来，代码进入一个无限循环，通过cap对象读取视频的每一帧。如果读取到的帧为空，则跳出循环。若读取到帧，则将每一帧图像转换为灰度图像，并进行二值化、腐蚀和膨胀操作，以便进行边缘检测。接着，定义四个关键点：左下角、右下角、左顶点和右顶点，用于ROI。通过调用region\_of\_interest函数，提取ROI内的图像。之后，使用HoughLinesP函数进行霍夫变换，检测出ROI内的直线。然后，使用fit\_lines函数对检测到的直线进行拟合，分别得到左车道线和右车道线的参数。如果找到了左车道线，代码更新last\_left\_line变量的值。

接下来，通过intersection\_point函数计算左右车道线的交点。并且在原图上绘制拟合的直线交点以下的部分。通过计算斜率和截距，确定左车道线的两个端点，并使用line函数绘制直线。

最后，代码显示处理后的图像，将帧写入视频文件，并检测是否按下了Esc键来结束程序。释放VideoCapture对象，关闭所有窗口。

**4、运行结果**

****

**5、总结**

该程序主要实现了使用OpenCV对道路上的车道线进行检测和拟合，涉及以下技术：

1）图像处理：灰度转换、二值化、腐蚀、膨胀、边缘检测；

2）几何计算：拟合直线、计算交点；

3）霍夫变换：检测直线。

实现该程序的主要难点有：

1）车道线拟合：使用最小二乘法对检测到的点进行直线拟合；

2）检测到的直线筛选：根据斜率判断是左车道线还是右车道线。

并且由于视频按帧处理会导致左侧不连续的车道线无法拟合出完整的直线，基于此，我进行了相关优化：当该帧检测不到车道线时使用上一帧的结果输出。这样可以使车道线在视频中持续存在而不会出现某一帧没有直线输出。同时由于拟合直线之后直接输出会输出完整直线，即超过车道线的部分也会输出，因此我通过计算两条直线的交点并输出交点以下直线的部分来解决了这个问题。

当然受制于技术本身的限制，以上方案仅能实现对直线车道线的处理，并且对光照和视角变换的鲁棒性不强，因此后续可针对上述不足进行进一步改进。