**车道线检测**

**汇报总结**

**首都师范大学**

**信息工程学院**

**四组**

**张守岐**

**1201002037**

目录

[一、 实验说明 2](#_Toc20113)

[二、 简易版车道线识别 2](#_Toc23507)

[三、 较先进车道线识别 3](#_Toc20982)

[四、 问题总结 5](#_Toc3080)

* 1. **实验说明**

1. 说明

首先，这次项目我们组一开始选的算法太过简单，在后期更换了代码。但因为太难和缺少人手，看不懂部分算法，导致我们没在代码层面上下手。

1. 车道线研究目的和意义

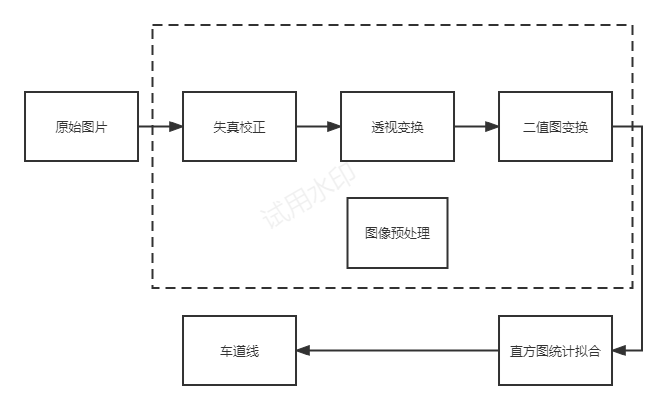
车道线检测是一种基础且重要的视觉任务，对于自动驾驶、交通管理和驾驶员辅助系统具有重大意义。车道线检测的目的是准确地检测和提取道路上的车道线，并为车辆提供定位和路径规划的依据。

车道线检测主要应用于自动驾驶，完成对车道线的识别后，自动驾驶系统就可以实现车辆横向运动的主动安全功能和控制功能。

车道线检测可以确定当前车辆与车道的位置关系，在车辆偏离车道时，系统就可以获知并通过声音、触觉等方式来提醒驾驶员，避免在车辆越线后，触发潜在的横向碰撞或其他风险，也就是车道偏离预警。同样还可以做到车道居中辅助、自动变道辅助等。

* 1. **简易版车道线识别**

1. 算法结构



失真校正：车载摄像头存在失真情况，即照片的中部拍的会比较近，四周拍的会比较远。要解决这个问题，需要用cv2.findChessboardCorners()来对相关摄像头的棋盘格使用，获得相关参数后使用cv2.calibrateCamera()函数，获得相机的畸变系数。把畸变系数导入cv2.undistort()函数，就能校正图片。

透视变换：通过cv2.getPerspectiveTransform()函数，把车道线变成俯视图片。

直方图统计拟合：就只统计灰度直方图，得出车道线。

上面算法都是最简单的算法，存在很多问题。比如失真校正需要提前得到相机的棋盘格来校正图片；透视变换只在固定区域内透视变换，只检测这个区域内的车道线检测，太被动而且不适配实际需求；得出车道线的算法无法在很多情况下生效，在大规模处理图片时速度也不行。

1. 实验结果

如下图所示，该算法只在意外和干扰最少的情况下生效。



因为很多图跑不出来，所以没有跑评价指标。

1. 实验总结

传统的方法还是不行，比不上深度学习。在这一块领域上深度学习的确在效率上远超传统方法。

* 1. **较先进车道线识别**

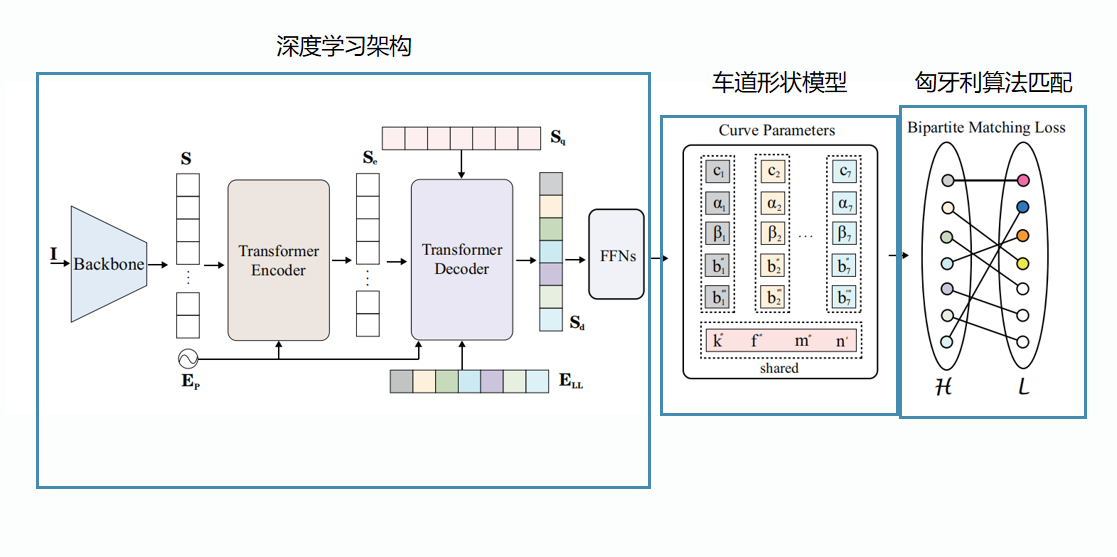
最新的车道线识别找不到代码，所以用的参考链接中的20年的车道线识别。

1. 研究背景

早期的方法通常首先生成分割结果，然后采用后处理，如分段聚类和曲线拟合。这些方法效率低下，在学习分割时忽略了全局上下文。为了解决上下文学习问题，一些方法使用消息传递或额外的场景注释来捕获全局上下文以增强最终性能，但这些方法不可避免地消耗更多的时间和数据成本。与这些方法不同，基于软注意力的方法生成一个空间加权图，在没有外部消耗的情况下提取出更丰富的上下文。然而，加权映射只测量特征的重要性，限制了它的使用来考虑支持推断细长结构的特征之间的依赖关系。

另一方面，为了提高算法的效率，在目标检测中将管道转移到检测车道，无需上述分割程序和后处理，但它依赖于复杂的锚定设计选择和额外的非最大抑制，使得其比大多数车道检测器更慢。近年来，一种方法将该任务通过多项式回归重新定义为车道标记拟合，取得了显著的效率，但由于忽略了对全局上下文的学习，与其他方法仍存在较大的精度差距。

1. 算法结构



深度学习架构：由Backbone提取特征，训练后得到用于预测车道曲线参数的FFNs。深度学习网络络是用变压器块构建的，属于这个算法中较大的改进。

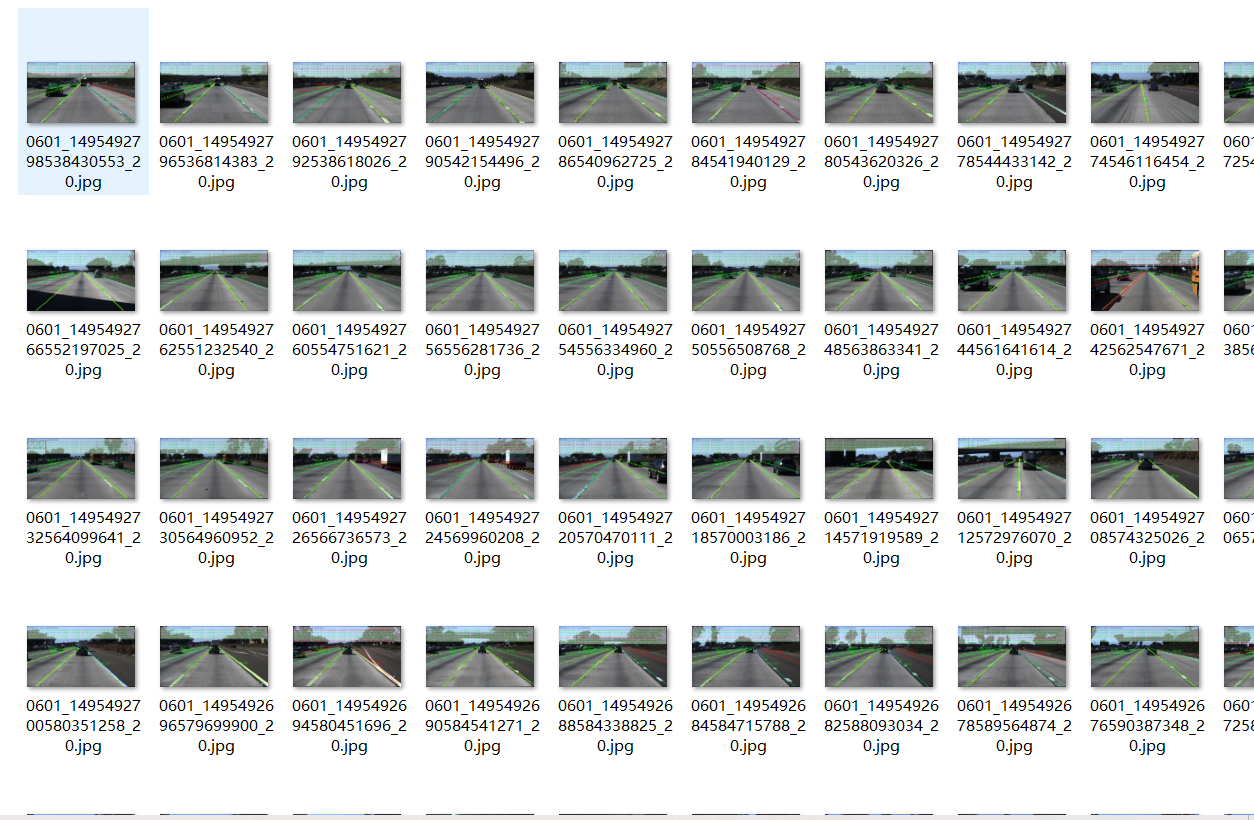
车道形状模型:因为全世界车道的弯曲曲线函数都是固定的，可以被拟合成的多项式，其中k, m, n, b都是实数。在倾斜相机的光轴与地面成一个角度的情况下，曲线从相机指向倾斜图像平面的曲线就可以用来表示，其中，k’, m’, n’, b’是参数和相机内在参数和外在参数组合得出的实数，（u，v）是图像平面上的一个像素，ϕ为光轴与地面的角度。

匈牙利算法匹配：采用匈牙利算法对预测参数和地面真实车道之间进行匹配。

算法整体采用端到端的匹配模式，重视全局信息的学习，所有多项式的参数都有具体的物理意义，所以在预测极端情况时会有优势。

1. 实验结果

我们使用了TuSimple车道检测数据集的最初版本来对算法进行评估。TuSimple数据集由6408张带注释的图像组成，包括一个3268的训练集、358的验证集和2782的测试集。它们是由1280\*720的前视图摄像机在美国高速公路上的各种交通和天气条件下录制的。



得到实验结果如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **方法** | **Acc** | **FP** | **FN** | **FPS** |
| FastDraw | 95.20 | 0.0760 | 0.0450 | 90 |
| SCNN | 96.53 | 0.0617 | **0.0180** | 7 |
| ENet-SAD | 96.64 | 0.0602 | 0.0205 | 75 |
| PINet | 96.70 | **0.0294** | 0.0263 | 30 |
| 线-CNN | **96.87** | 0.0442 | 0.0197 | 30 |
| 实验结果 | 96.22 | 0.0286 | 0.0329 | 117 |

在精度（Acc）上，实验结果比最好的线-CNN方法低了0.67%，但相比较其他方法，精度仅比FastDraw方法高。

在假阳性(FP)上，实验结果效果最好。

在假阴性（FN）上，实验结果仅比FastDraw方法好，落后了最好的SNCC约82.8%。

所有实验结果在GTX1650下得出。

1. 实验结论

整个方法在需要最小的参数和运行时间消耗的同时，实现了更先进的车道检测性能。同时，这种方法能够更可靠地适应数据集的变化，也更容易部署在移动设备上。同时，实验存在假阴性过高的问题，需要继续改进。

* 1. **问题总结**

1.环境搭建

在最开始，我在pytorch的环境搭建上遇到了困难。主要由三个：版本的不统一、包的安装中断和代码没标出的环境搭建步骤。包的安装和没标出的环境需求这两个问题在我查找后解决。对于第一个问题，其实后面的代码是我们采用的第三个代码，第二个代码因为这个问题，当我尝试了一天都没法解决后，就扔了。那一个代码是18年的，只能说较先进，但我论文都看了一部分了，只当补充算法知识了。

2.小组相关

首先，两个人人手完全不够。有事情，比如生病时，工作整个就停了。在算法和代码的交流时效率也不高，下次做项目要多拉点人。

3.总结

这次小组合作存在很多问题，我需要思考解决方法。算法方面要实时关注最新的论文，了解最新的思考方式。

车道线识别这一块领域还有很大的发展空间和要求。