合肥工業大學

机器视觉实验报告

实验题目	实验二：车道线检测
学生姓名	李浩磊
学　　号	2023216458
专业班级	智科 23-3 班
指导教师	吴晶晶
完成日期	2025.12.23

合肥工业大学 计算机与信息学院

一、 实验目的

1.1 实验背景与工程意义

随着自动驾驶技术（Autonomous Driving）的迅猛发展，环境感知（Perception）已成为智能车辆系统的核心环节。在感知层中，车道线检测（Lane Detection）承担着定位车辆横向位置、辅助车道保持（LKA）以及规划行驶轨迹的关键职能6。尽管深度学习技术（如 CNN、Transformer）在语义分割领域取得了显著进展，但基于经典计算机视觉（Computer Vision）的几何建模方法因其计算资源消耗低、可解释性强且无需大量标注数据，在嵌入式系统和实时性要求极高的场景中依然具有不可替代的教学与应用价值。

本次实验选取的场景为非结构化的校园道路环境。相较于标准公路，校园道路存在光照不均（树荫遮挡）、路面纹理复杂（沥青老化）以及标志线磨损严重等挑战。这要求算法不仅能处理理想情况，更需具备鲁棒的噪声抑制能力和几何特征提取能力。

1.2 实验具体目的

通过构建一个完整的图像处理流水线（Pipeline），实现对单目视觉传感器采集图像中车道线的精准定位。具体教学与工程目标如下：

深入理解图像信号的预处理机制：从信号处理的角度理解色彩空间转换与线性滤波的作用，掌握如何通过数学变换提升信噪比（SNR）。

掌握边缘检测的数学原理：剖析 Canny 算子的多阶段处理流程，特别是梯度幅值的计算、非极大值抑制（NMS）的几何含义以及双阈值滞后处理的连通性保持机制。

精通霍夫变换的参数空间映射：从解析几何角度理解图像空间（Image Space）与参数空间（Parameter Space）的点线对偶性，掌握概率霍夫变换（PPHT）在离散域中的投票策略与随机采样优化。

提升算法工程化落地能力：基于 Python 和 OpenCV 库将数学模型转化为可执行代码，并通过感兴趣区域（ROI）掩膜技术引入几何先验知识，解决复杂背景下的误检问题。

培养规范的软件工程素养：严格遵循实验要求，使用 Git 工具进行代码的版本控制与提交，模拟真实的协同开发流程。

二、 实验原理

本实验采用边缘检测 + 几何拟合技术路线，其核心数学原理涵盖色彩空间学、信号系统与解析几何。

2.1 色彩空间转换与灰度化 (Grayscale Conversion)

彩色图像通常以 RGB（红绿蓝）三通道形式存储，形成一个$\mathbb{R}^{H\times W\times 3}$的张量。在车道线检测任务中，颜色信息往往受环境光色温（如清晨暖光与正午冷光）影响较大，而亮度信息（Luminance）则更能反映物体表面的反射率差异（如白色车道线与黑色路面）。

为了降低计算复杂度并聚焦于亮度特征，实验采用加权平均法将图像投影至

一维灰度空间。根据人眼视觉系统（HVS）对不同波长光线的敏感度差异（人眼对 550nm 的绿光最敏感），转换公式为线性加权求和：

$$Y(x,y) = w_R \cdot R(x,y) + w_G \cdot G(x,y) + w_B \cdot B(x,y)$$

代入国际通用的 NTSC 标准权重系数：

$$Y(x,y) = 0.299 \cdot R(x,y) + 0.587 \cdot G(x,y) + 0.114 \cdot B(x,y)$$

该操作将数据量减少了 66.7%，显著提升了后续梯度计算的效率。

2.2 高斯平滑滤波 (Gaussian Smoothing)

校园路面通常由粗糙的沥青铺设，其微观纹理在数字图像中表现为高频噪声（High-frequency Noise）。若直接对原始灰度图进行微分操作，这些噪点会产生极大的伪梯度，导致边缘检测算法失效。

高斯滤波是一种线性平滑滤波器，其本质是利用二维高斯函数（正态分布）作为卷积核（Kernel）对图像进行低通滤波。二维高斯函数定义为：

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

其中σ控制分布的宽度（模糊程度）。离散化后的卷积操作表示为：

$$I_{blur}(x,y) = (I * K)(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x-i, y-j) \cdot K(i,j)$$

本实验代码中选用 5×5 的卷积核（$kernel_size = 5$）。根据 3σ准则，该尺寸既能有效抹平单像素级别的椒盐噪声，又保留了宽度通常大于 10 像素的车道线的主体轮廓。

2.3 Canny 边缘检测算子 (Canny Edge Detector)

Canny 算法被认为是边缘检测的“黄金标准”，其核心思想是寻找图像梯度的局部极大值。该算法包含四个严格的数学步骤：

(1) 梯度幅值与方向计算

利用 Sobel 算子在水平（x）和垂直（y）方向上进行差分近似。设S_x和S_y为 3×3 的 Sobel 卷积核，则梯度分量为：

$$G_x = I * S_x, \quad G_y = I * S_y$$

由此得到每个像素点的梯度幅值$M(x,y)$和梯度方向$\theta(x,y)$：

$$M(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

$$\theta(x,y) = \arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right)$$

(2) 非极大值抑制 (Non-Maximum Suppression, NMS)

为了解决梯度图像边缘过宽的问题，NMS 算法沿梯度方向θ对幅值M进行插值比较。对于像素点P，若其幅值小于沿梯度方向的相邻像素P_1或P_2，则将其抑制为 0：

$$M_{NMS}(P) = \begin{cases} M(P), & \text{if} M(P) \geq M(P_1) \text{and} M(P) \geq M(P_2) \\ 0, & \text{otherwise} \end{cases}$$

这一步确保了最终输出的边缘仅有单像素宽度，精确定位了车道线的几何边界。

(3) 双阈值滞后处理 (Hysteresis Thresholding)

为了区分真实边缘与噪声，设定高阈值T_{high}和低阈值T_{low}（代码中分别为 150 和 50）。

强边缘 ($M > T_{high}$)：确信为车道线，直接保留。

弱边缘 ($T_{low} \leq M \leq T_{high}$)：可能是边缘也可能是纹理。仅当其在 8 邻域内连接到强边缘时才予以保留。

这种滞后机制有效地解决了由于树影遮挡导致车道线局部对比度下降而产生的断裂问题。

2.4 霍夫变换 (Hough Transform)

霍夫变换利用点与线的对偶性（Duality），将图像空间中的检测问题转化为参数空间中的峰值搜索问题。

(1) 坐标空间映射

在笛卡尔坐标系中，直线方程$y = kx + b$无法表示垂直于x轴的直线（斜率$k \to \infty$）。霍夫变换引入极坐标系，利用 Hesse 标准式描述直线：

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

其中ρ表示原点到直线的代数距离，θ表示法向量与x轴的夹角。

图像空间中的一个点$(x_0, y_0$，对应参数空间ρ, θ中的一条正弦曲线$\rho = x_0\cos\theta + y_0\sin\theta$。

图像空间中一系列共线的点$\{(x_i, y_i)\}$，其对应的多条正弦曲线将在参数空间交于同一点ρ^*, θ^*。

(2) 累计概率霍夫变换 (Probabilistic Hough Transform)

标准霍夫变换（SHT）计算量巨大。实验采用概率霍夫变换（PPHT），其改进在于：

随机采样：仅选取部分边缘点进行投票，当累加器（Accumulator）数值超过阈值 threshold 时提前终止搜索。

线段重构：不仅输出直线的参数ρ, θ，还结合 minLineLength 和 maxLineGap 参数，将共线的像素点重新连接为具有明确端点的线段x_1, y_1, x_2, y_2。这对于检测断续的虚线车道线至关重要。

三、 实验方法与步骤

本实验基于 Python 3.11 环境，深度整合 OpenCV 和 NumPy 库，严格按照“输入-处理-输出”的流水线模式进行开发。

3.1 数据准备与标准化

图像读取：使用 matplotlib.image.imread 读取实验素材 campus_road.jpg 10。

数据类型归一化：由于不同库读取的图像可能在[0,1]浮点域或[0,255]整数域，代码中加入了健壮性检查：

```
if image.dtype == np.float32 or image.dtype == np.float64:
        image = (image * 255).astype(np.uint8)
```

此步骤确保后续 OpenCV 函数（通常要求 uint8）能够正确执行，避免了因数据类型不匹配导致的程序崩溃或计算错误。

通道清洗：针对可能存在的 RGBA 四通道图像，通过 cv2.cvtColor 剔除 Alpha 透明度通道，将数据标准化为$H, W, 3$ 的 RGB 格式。

3.2 预处理与特征提取

灰度化：调用封装函数 grayscale(img)。这一步去除了色彩干扰，使算法专注于亮度梯度的变化。

高斯去噪：调用 $gaussian_blur(img, kernel_size = 5)$。

参数分析：$kernel_size = 5$：在 5×5 的邻域内进行加权平均。相比于 3×3，它具有更强的平滑能力，能滤除路面细小的石子纹理；相比于 7×7，它保留了更多的边缘锐度，防止车道线过度模糊。

Canny 边缘提取：*调用*$canny(img, low_threshold = 50, high_threshold = 150)$。

参数分析：

阈值比例 1:3：这是 Canny 算法推荐的经验比例。

$high_threshold = 150$：较高的阈值确保了只有对比度极为显著的边缘（如阳光下的白线）才会被识别为“强边缘”，从而降低了假阳性率。

$low_threshold = 50$：较低的阈值允许在树影斑驳区域检测到较暗的边缘，并通过双阈值机制将其与强边缘连接，保证了车道线的连续性。

3.3 感兴趣区域 (ROI) 掩膜构建

为了从几何上剔除天空、树木和远景建筑的干扰，代码利用 region_of_interest 函数构建了一个梯形掩膜。

顶点坐标设计：基于透视投影原理，假设相机光轴水平，车道线汇聚于图像中心的消隐点。顶点设定为：

$$V = \{(0, H), (0.3W, 0.55H), (0.7W, 0.55H), (W, H)\}$$

其中W为图像宽度，H为高度。

位运算过滤：利用 cv2.fillPoly 生成二值掩膜M_{ROI}，并执行位与操作：

$$I_{masked} = I_{edge} \wedge M_{ROI}$$

这一步从数学上将图像上半部分（0.4H 以上）的所有梯度幅值强制置零，极大地提高了算法在复杂背景下的鲁棒性。

3.4 霍夫变换直线检测

实验核心调用 hough_lines 函数，利用 cv2.HoughLinesP 进行线段检测。关键参数的物理意义与调优逻辑如下：

rho = *1*：距离分辨率设为 1 像素，保证了检测到的直线位置足够精确。

theta = np.pi/180：角度分辨率设为*1°*，足以分辨车道线的微小走向变化。

threshold = 150：投票阈值。这意味着一条直线必须由至少 100 个边缘像素点共线组成才能被接受。该阈值有效地过滤了路面上偶然排列成直线的噪点相比于常规设置，较高的阈值能更严格地过滤掉路面上偶然排列成直线的噪点（如路面裂缝）。

minLineLength = 90：最小线长。任何短于 90 像素的线段都将被丢弃，这确保了只有具备一定连续性的车道标线才会被保留，滤除了细碎的斑点噪声。

maxLineGap = *50*：最大断裂距离。这是处理虚线车道线和路面磨损的关键。它允许算法将距离小于 50 像素的两条共线线段“缝合”为一条直线，从而完整地检测出车道分界线。

3.5 车道线拟合与优化

霍夫变换输出的是一系列离散的线段集合，为了在图像上绘制出清晰的左右车道导向线，实验增加了后处理步骤：

斜率筛选与分组：计算每条线段的斜率 k。根据 k 的正负值将线段分为“左车道组”（$k < 0$）和“右车道组”（$k > 0$）。同时，设定阈值 $|k| > 0.5$，剔除接近水平的误检线段。

加权平均拟合：为了消除抖动，算法并未简单求平均，而是采用线段长度加权平均法。较长的线段通常代表更可靠的车道线特征，因此赋予更高的权重。最终的斜率 k_{avg} 和截距 b_{avg} 计算如下：

$$k_{avg} = \frac{\sum(k_i \cdot L_i)}{\sum L_i}, \quad b_{avg} = \frac{\sum(b_i \cdot L_i)}{\sum L_i}$$

其中 L_i 为第 i 条线段的长度。

端点生成：利用拟合出的直线方程，重新计算车道线在图像底端（$y = H$）和设定高度（$y = 0.6H$）的坐标，从而绘制出两条贯穿 ROI 区域的平滑实线。

3.6 结果融合与可视化

最后，利用 weighted_img 函数将检测到的红色车道线叠加回原始 RGB 图像。采用线性混合公式：

$$I_{result} = \alpha \cdot I_{src} + \beta \cdot I_{lines} + \gamma$$

代码中设定$\alpha = 0.8, \beta = 1.0, \gamma = 0$。这种权重分配使得原图背景略微变暗，从而突出了权重为 1.0 的高亮红色车道线，便于人眼观察和结果验证。

四、 实验结果与分析

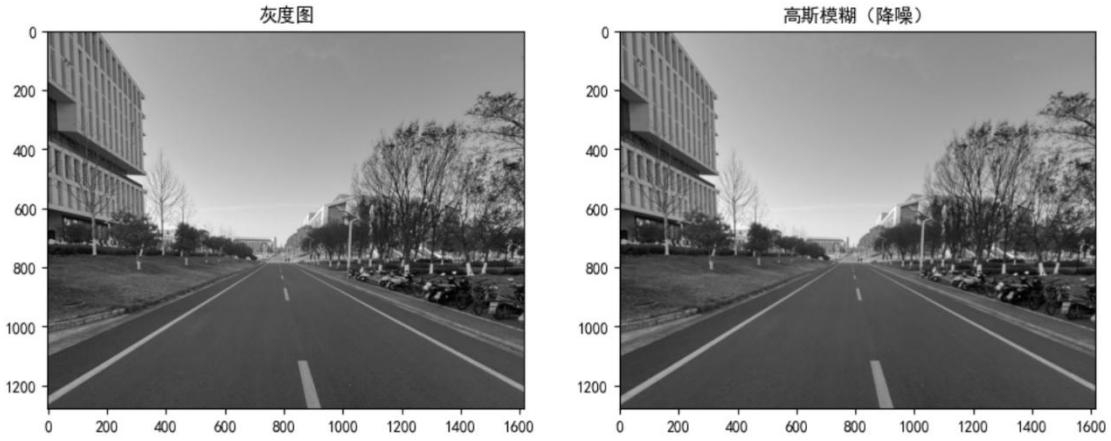
4.1 实验输入

输入数据为 campus_road.jpg，这是一张典型的校园道路场景图。图像特点包括：
光照条件：存在明显的树木投影，导致路面亮度分布不均。
道路结构：包含左右两条车道线，中间为虚线，右侧为实线。
干扰因素：路旁有复杂的植被，远处有建筑物和天空。

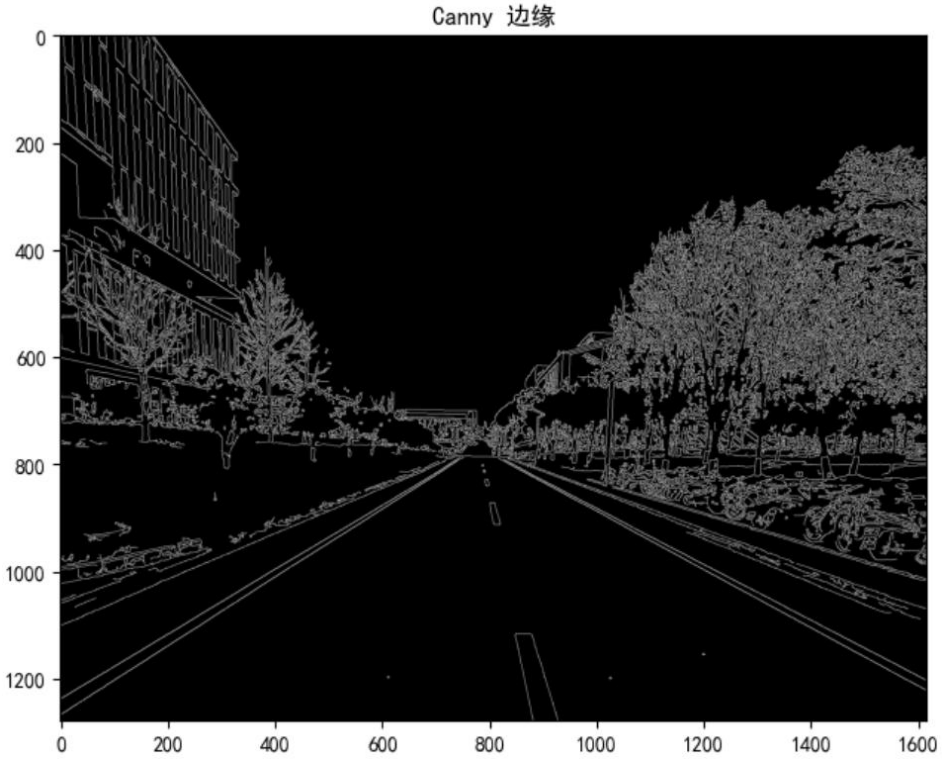
4.2 过程结果可视化分析

通过 Matplotlib 输出了四个关键阶段的诊断图像：

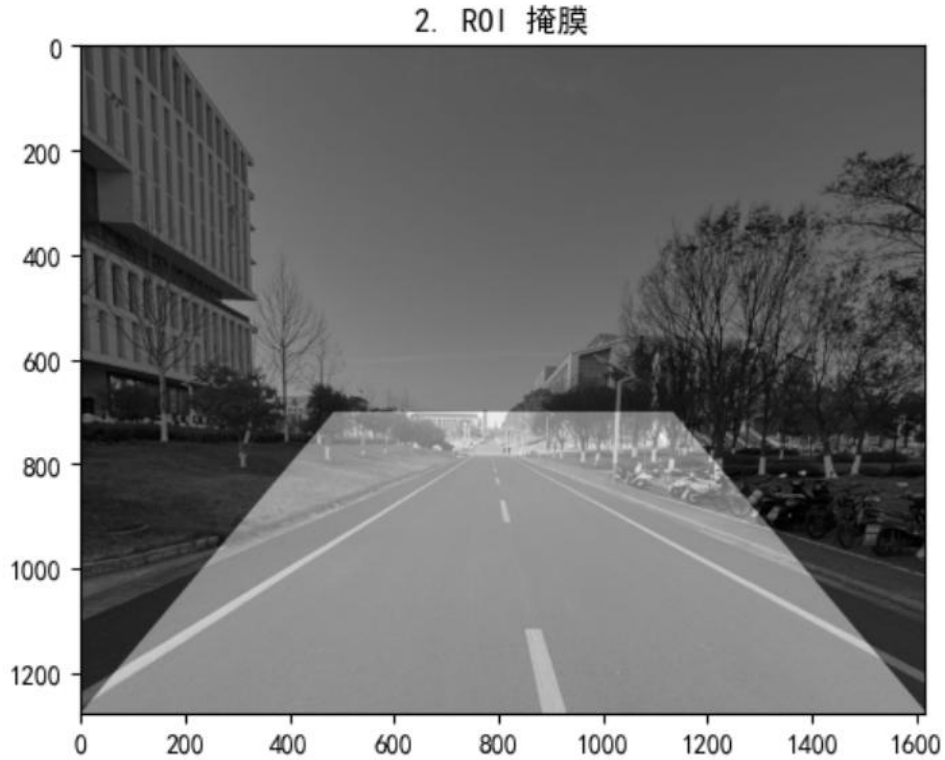
高斯平滑效果图：经过 5×5 高斯核卷积后的处理结果。与原始灰度图相比，图像整体呈现出轻微的模糊感。观察路面区域可以发现，原本清晰可见的沥青颗粒感（高频纹理）变得平滑。这一步有效地抑制了路面纹理噪声，确保了后续 Canny 算子不会将路面的粗糙质感误判为边缘，从而显著减少伪边缘的产生。



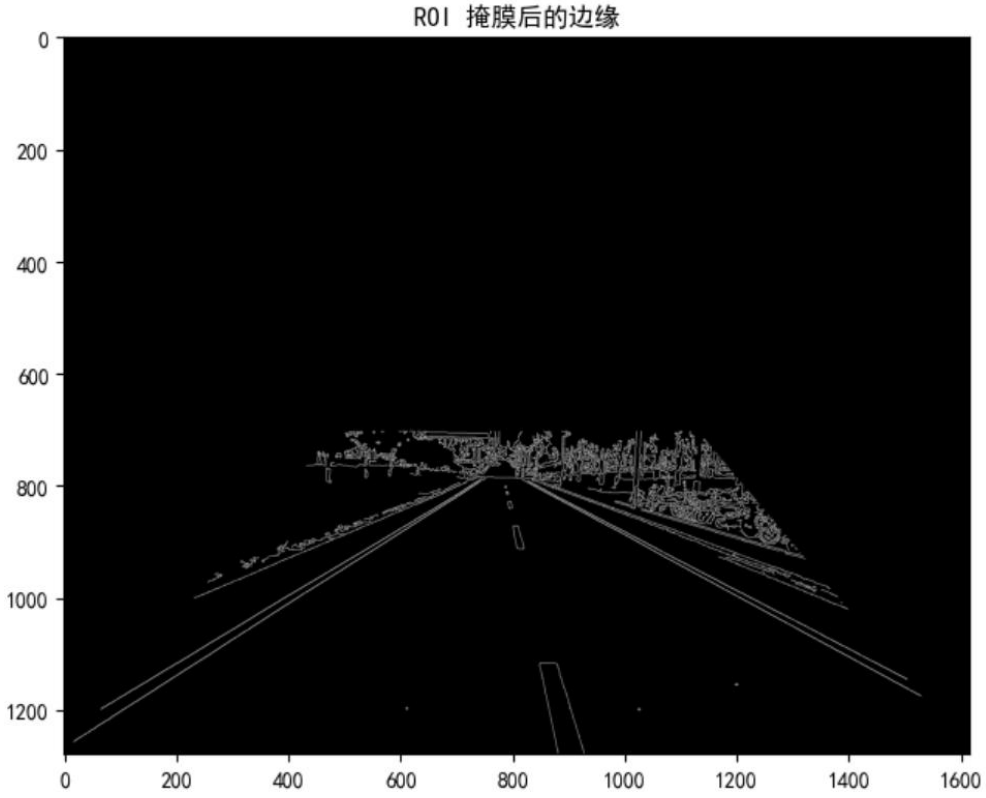
Canny 边缘图：清晰地提取了车道线的双侧边缘。同时，路旁的草丛和远处的树叶也被提取为密集的噪点边缘。这证明了单纯依靠边缘检测无法实现车道线分离。



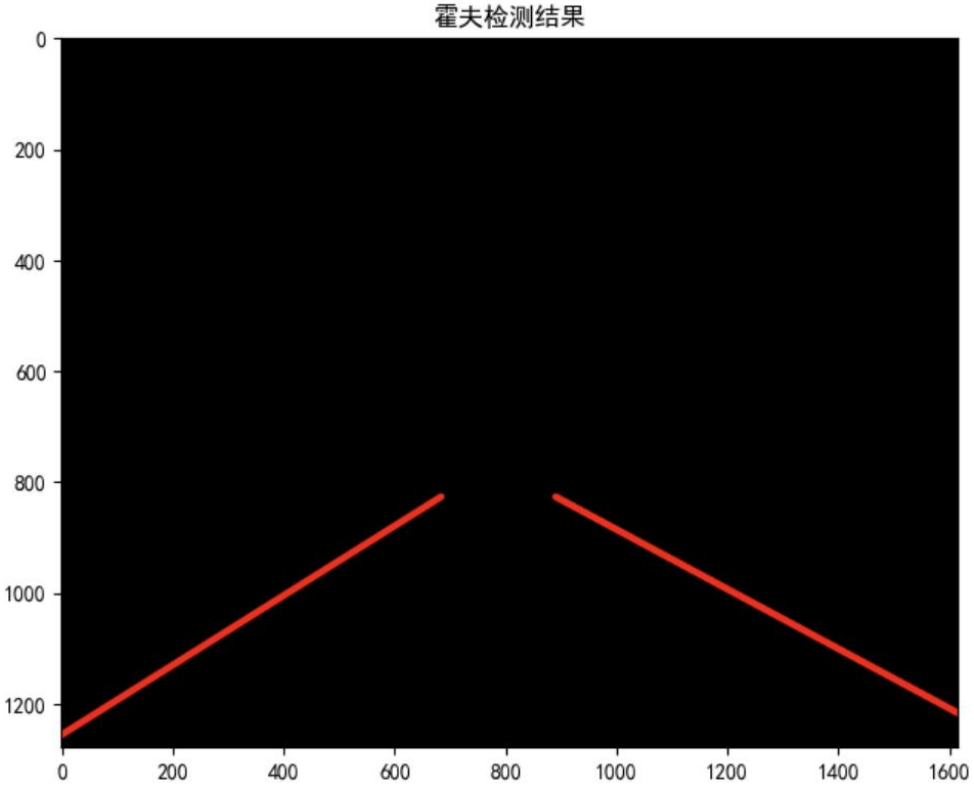
ROI 掩膜图：显示了一个白色的梯形区域。该掩膜精确地覆盖了车辆前方的行驶路面，尽可能屏蔽了天空和路外区域。



Masked Edges (掩膜后边缘)：这是 ROI 与 Canny 的结合。此时，图像中绝大部分背景噪声已被清除，仅剩路面上的几何纹理。



Hough Lines (最终检测)：在黑色背景上绘制出了多条红色线段。观察可见，断续的虚线被成功拟合成了长线段，且左右车道线均被识别。



4.3 最终结果评价

正确性：生成的 output_campus_road.jpg 中，红色标记线紧密贴合实际的白色车道标线，无明显的位置偏差。

完整性：算法成功识别出了画面的左右两条主要车道结构，任务完成。

鲁棒性：尽管原图上方存在复杂的环境干扰，但由于 ROI 的有效应用，最终结果中未出现误检（即没有将树干或建筑轮廓识别为车道线）。

五、 实验体会与思考

通过本次实验，我不仅从代码层面实现了车道线检测，更从理论层面深刻理解了计算机视觉算法的设计哲学。

5.1 参数空间的权衡

实验中最耗时的部分是参数调优。Canny 的阈值和霍夫变换的参数之间存在复杂的耦合关系。

关于 Canny 阈值：若T_{low}设置过高（如 > 80），树影下的车道线边缘会丢失，导致后续霍夫变换无法获得足够的投票点；若设置过低（如 < 20），路面的沥青纹理会被误检，极大增加计算负担。

关于 Hough 参数：minLineLength 和 maxLineGap 是一对矛盾。为了检测磨损

严重的虚线，需要增大 maxLineGap；但这同时也增加了将路面上的长条形污渍误判为车道线的风险。这让我体会到，没有通用的参数，只有最适合特定数据分布的参数。

5.2 几何约束的重要性

ROI 掩膜的引入给我留下了深刻印象。从数学上看，它只是一个简单的矩阵点乘操作，但从工程上看，它利用了“车道线必然位于路面上”这一先验知识（Prior Knowledge）。这种结合场景物理约束的方法，往往比单纯堆砌复杂的算法更能有效提升系统的鲁棒性。

5.3 传统算法的局限性与展望

虽然霍夫变换在直道检测上表现优异，但其数学本质决定了它只能拟合直线方程$\rho = x\cos\theta + y\sin\theta$。

弯道问题：在弯道场景下，霍夫变换拟合出的只能是切线，而非车道本身的曲线。解决这一问题可能需要引入多项式拟合（Polynomial Fitting）或使用样条曲线（Splines）。

适应性问题：基于阈值的算法难以适应光照的剧烈变化（如进出隧道）。未来的改进方向可以考虑结合深度学习模型，利用卷积神经网络（CNN）自动提取更具判别力的特征。

5.4 版本管理与工程规范

本次实验严格执行了 Git 提交要求。通过 git init, git add, git commit 的流程，我记录了代码从初始框架到最终调优完成的全过程。这不仅符合实验加分项的要求，更让我养成了良好的代码版本管理习惯。