合肥工業大學

（计算机与信息学院）

机器视觉实验报告

专　业　班　级　智能科学与技术 21-1 班

学 生 姓 名 学 号　郭宇童 2021214582

任　课　教　师　洪日昌

实　验　名　称　实验 2 霍夫变换车道线检测

2023-2024 学年第 1 学期

摘要：

车道线检测是自动驾驶系统中的关键技术之一，它为车辆在道路上的安全行驶提供了必要的视觉信息。正确地识别和跟踪道路上的车道线对于保持车辆在车道内、避免交通事故以及辅助驾驶决策至关重要。在计算机视觉领域，多种算法和技术已被开发用于车道线的检测和识别，其中霍夫变换是一种经典且广泛使用的方法。

霍夫变换是一种特征提取技术，用于在图像中识别复杂模式的简单几何形状，如直线、圆等。在车道线检测的背景下，霍夫变换被用来识别和提取图像中的直线特征，这些直线特征通常对应于车道线。霍夫变换的基本思想是将图像空间中的点映射到参数空间，并在参数空间中寻找满足特定形状条件的参数集合。

本文旨在探讨如何使用霍夫变换来检测图像中的车道线。实验将以自拍摄的校园道路图像为基础，目标是在图像中准确识别车道线的位置。文章将从霍夫变换的基本原理讲起，解释其如何应用于车道线的检测，并展示在实际图像上的应用效果。本文还将提供实现霍夫变换的详细算法和编程技巧，包括如何处理图像、提取边缘以及如何应用霍夫变换来识别直线特征。

关键词：霍夫变换；车道线检测；自动驾驶；计算机视觉；特征提取；

引言

在数字图像处理和计算机视觉领域，识别和提取图像中的几何形状，尤其是直线和曲线，是一项基本且关键的任务。这一任务在自动驾驶、道路监控、机器人导航等多个应用领域中扮演着重要角色。霍夫变换是一种用于检测图像中简单形状如直线和圆的经典方法，由 Paul Hough 于 1962 年首次提出并后续被 Richard Duda 和 Peter Hart 在 1972 年通过霍夫变换的标准化而广泛传播。

霍夫变换的核心思想是将图像空间中的特征点映射到参数空间，以便更容易地识别特定的形状。特别是对于直线检测，霍夫变换通过一种投票机制在参数空间中累积交叉点，从而有效地识别和提取图像中的直线。这种方法的优势在于其能够从包含噪声和不完整特征的图像中准确检测出几何形状。

本实验旨在探索霍夫变换在直线检测和车道线识别方面的应用。通过在不同的道路图像上应用霍夫变换，我们将评估其在实际场景中的表现，特别是在处理不同类型的道路和多种环境条件下的效果。实验将涉及霍夫变换的实现细节，包括其在图像预处理、边缘检测后的应用，以及如何从参数空间中提取直线信息。

通过本实验，我们期望深入理解霍夫变换的工作原理及其在车道线检测等实际计算机视觉任务中的应用效果。我们还将探讨霍夫变换与其他形状检测方法的比较，以及它在不同应用背景下的优势和局限性。

1 霍夫变换简介

1.1 霍夫变换的数学原理

霍夫变换是一种用于检测图像中简单几何形状的技术，尤其是直线和圆。它的数学原理

基于将图像空间中的特征点映射到参数空间的思想。

1.1.1 直线的参数表示

在二维图像空间中，一条直线可以用方程 $y=mx+c$ 表示，其中 m 是斜率，c 是截距。然而，这种表示形式在处理垂直线时会遇到问题，因为垂直线的斜率为无穷大。

为了解决这个问题，霍夫变换采用极坐标系来表示直线。在极坐标系中，一条直线可以表示为 $r=x\cos\theta+y\sin\theta$，其中 r 是原点到直线的垂直距离，θ 是该垂线与x轴的夹角。

1.1.2 参数空间的映射

霍夫变换的关键在于将图像空间中的每个点转换到参数空间。对于图像中的每个点 (x,y)，我们考虑所有可能通过该点的直线，并将这些直线对应的 (r,θ) 值绘制在参数空间中。

在参数空间中，一组交叉的曲线可能对应于图像空间中的一个点，而一组交叉点可能表示图像中的一条直线。

1.1.3 累加与投票机制

在实际应用中，霍夫变换使用一个累加器（通常是二维数组），用于记录参数空间中每个 (r,θ) 单元的投票数。对于图像中的每个点，我们计算它所有可能的 (r,θ) 值，并在累加器的相应位置增加计数（即投票）。最终，拥有较高投票数的 (r,θ) 单元表示图像中存在的直线。

1.1.4 阈值判断与直线提取

通过设置一个适当的阈值，我们可以从累加器中选择具有最高投票数的 (r,θ) 值，这些值代表图像中最显著的直线。最后，根据这些 (r,θ) 值，我们可以在原始图像中绘制相应的直线。

1.1.5 总结

霍夫变换在数学层面上的正确性使其成为检测图像中特定形状的强大工具，特别是在处理含有噪声或断裂边缘的图像时。通过将图像空间中的点转换到参数空间，并利用累加和投票机制，霍夫变换能够有效地识别和提取图像中的直线和其他几何形状。

1.2 霍夫变换的应用举例

Sobel 算子作为一种流行的边缘检测工具，在图像处理和计算机视觉领域有着广泛的应用。以下是一些具体的应用实例：

表 1 霍夫变换的应用

应用	描述
车道线检测	检测和跟踪道路上的车道线，从而辅助车辆导航。
物体识别与定位	识别和定位图像中的直线和圆形对象
图像分割与重建	被用于分割与重建断裂的图像边缘。
机器人导航	被用于识别环境中的路径和障碍物

建筑和工程图像分析	用于分析建筑图纸和工程图，识别线条和形状

2 车道线检测实验的编程实现

2.1 图像预处理

- grayscale(image): 将图像转换成灰度图。这是因为边缘检测通常在灰度图上进行，以降低计算复杂性。
- gaussian_blur(image, kernel_size): 使用高斯滤波器模糊图像。这有助于减少图像噪声，进而在后续的边缘检测中减少误检。
- canny(image, low_threshold, high_threshold): 应用 Canny 边缘检测算法。这个方法检测图像中的强边缘（高梯度区域）

```
# 灰度图转换
def grayscale(image):
    return cv.cvtColor(image, cv.COLOR_RGB2GRAY)
```

```
# Canny边缘检测
def canny(image, low_threshold, high_threshold):
    return cv.Canny(image, low_threshold, high_threshold)

# 高斯滤波
def gaussian_blur(image, kernel_size):
    return cv.GaussianBlur(image, (kernel_size, kernel_size), 0)
```

图 图像预处理

2.2 定义感兴趣区域并应用霍夫变换

- region_of_interest(image, vertices): 定义一个掩模以遮盖图像的非感兴趣区域。在这个案例中，只关注包含车道线的道路部分。
- hough_lines(img, rho, theta, threshold, min_line_len, max_line_gap): 实现霍夫变换来检测图像中的直线。这个函数的参数用于定义霍夫空间的精度和直线的选择标准。

```
def region_of_interest(image, vertices):

    mask = np.zeros_like(image)  # 生成图像大小一致的zeros矩

    # 填充顶点vertices中间区域
    if len(image.shape) > 2:
        channel_count = image.shape[2]
        ignore_mask_color = (255,) * channel_count
    else:
        ignore_mask_color = 255

    # 填充函数
    cv.fillPoly(mask, vertices, ignore_mask_color)
    masked_image = cv.bitwise_and(image, mask)
    return masked_image
```

图 定义感兴趣区域

```python
def hough_lines(img, rho, theta, threshold, min_line_len, max_line_gap):

    # rho: 线段以像素为单位的距离精度
    # theta : 像素以弧度为单位的角度精度(np.pi/180较为合适)
    # threshold : 霍夫平面累加的阈值
    # minLineLength : 线段最小长度(像素级)
    # maxLineGap : 最大允许断裂长度
    lines = cv.HoughLinesP(img, rho, theta, threshold, np.array([]), minLineLength=min_line_len, maxLi
    return lines
```

图 定义霍夫变换应用

2.3 绘制车道线并且融合图像

- draw_lines(image, lines, color, thickness): 这个函数用于在图像上绘制检测到的车道线。它将直线分为左右两组，根据斜率和位置判断，然后绘制代表车道的线条。
- weighted_img(img, initial_img, a, b, c): 将检测到的车道线图像与原始图像融合。这样做可以直观地显示车道线在原始道路图像中的位置。

3 实验运行结果

原始的图像如下所示：

选取图像为网上和校园内的图片各一张。

图 处理的源图片 1

图 处理的源图片 2

经过处理的图像如下所示：

图 车道检测后的图像

校园内图像处理后为：

可以看到直线并不理想，猜想是因为背景过于复杂。这个是我又写了一个 single.py,仅仅用于测试单条车道线检测任务的可行性。

图 校园内的图像检测

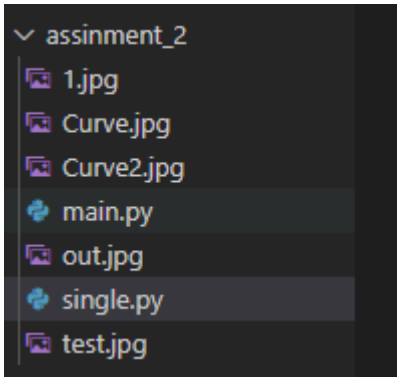

图 assignment2 的文件夹内容（main 为两条车道线检测，single 为单检测）