

**机器视觉课程实验报告**

**学生姓名**  褚周涛

**学 号**  2023217624

**专业班级**  智科3班

**指导教师**  吴晶晶

**完成日期**  2026/1/15

**合肥工业大学 计算机与信息学院**

**实验一 图像滤波**

**一、实验目的**

1.掌握图像卷积的基本原理和手动实现方法，理解卷积核在图像特征提取中的作用

2.实现Sobel边缘检测算子，理解梯度计算在边缘提取中的物理意义

3.实现颜色直方图的手动计算与可视化，掌握图像颜色分布分析方法

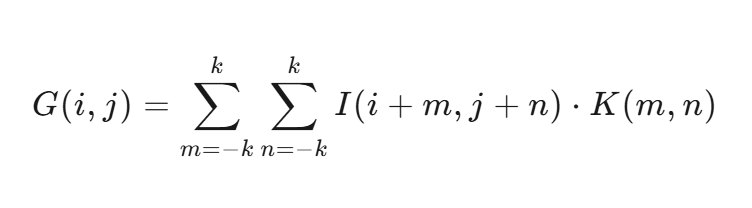
4.实现LBP纹理特征提取，理解局部二值模式在纹理描述中的应用

5.对比手动实现与OpenCV库函数的差异，加深对底层算法的理解

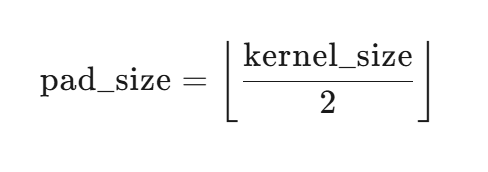
**二、实验原理**

**2.1 手动卷积实现原理**

卷积操作是图像处理的核心运算：



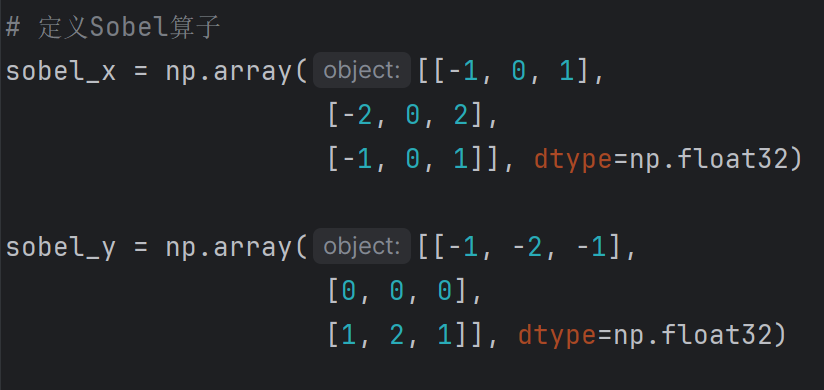
其中I 为输入图像，K 为卷积核，G 为输出图像。为保证输出尺寸一致，需进行边界填充（Zero Padding）：



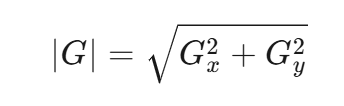
代码实现采用三层嵌套循环：外层遍历图像像素，内层遍历卷积核，通过np.pad实现零填充，最终使用np.clip限制像素值范围。

**2.2 Sobel边缘检测原理**

Sobel算子基于一阶导数近似，通过两个3×3卷积核分别计算水平和垂直方向梯度：



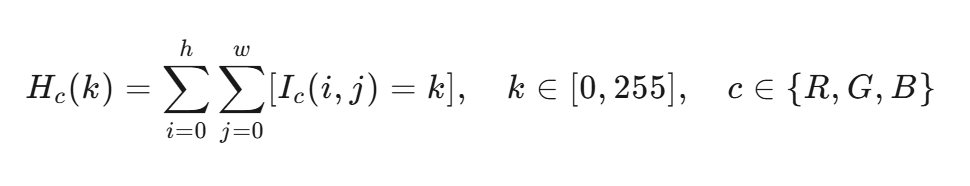
梯度幅值计算为：



该计算对噪声具有鲁棒性，且能突出水平和垂直边缘。

**2.3 颜色直方图原理**

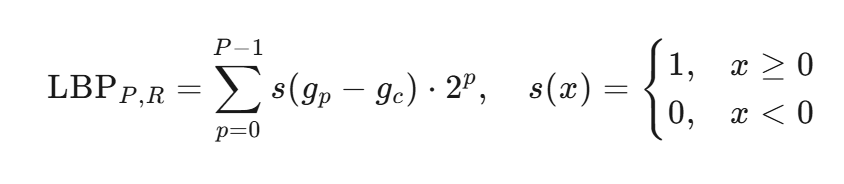
直方图统计每个颜色通道的像素分布：



def manual\_color\_histogram代码通过np.sum(channel == i)实现高效统计，避免了显式循环。

**2.4 LBP纹理特征原理**

局部二值模式将每个像素与其圆形邻域比较：



本实验采用P=8,R=1 的均匀模式，生成256维特征向量，对光照变化具有不变性。

**三、实验方法**

**3.1 系统架构设计**

图像输入 → 手动卷积模块 → Sobel滤波 → 给定核滤波 → 直方图分析 → LBP特征提取 → 结果输出

**3.2 核心算法流程**

**手动卷积流程：**

1.获取图像和卷积核尺寸

2.计算填充尺寸pad\_h = k\_h // 2

3.对每个通道：使用np.pad进行零填充

4.双重循环遍历像素，提取卷积区域并进行元素乘加运算

5.结果裁剪至[0,255]范围并转换为uint8

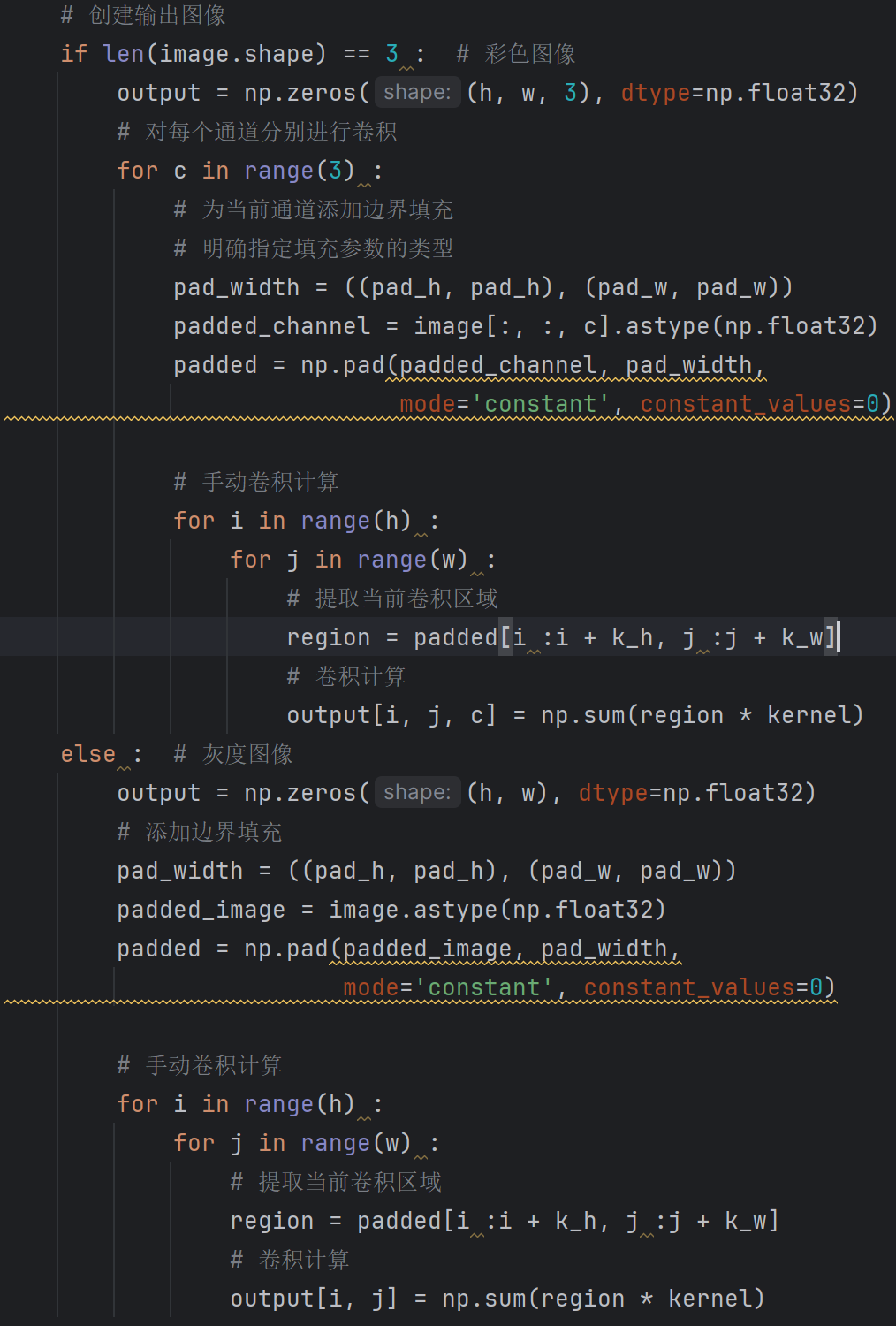
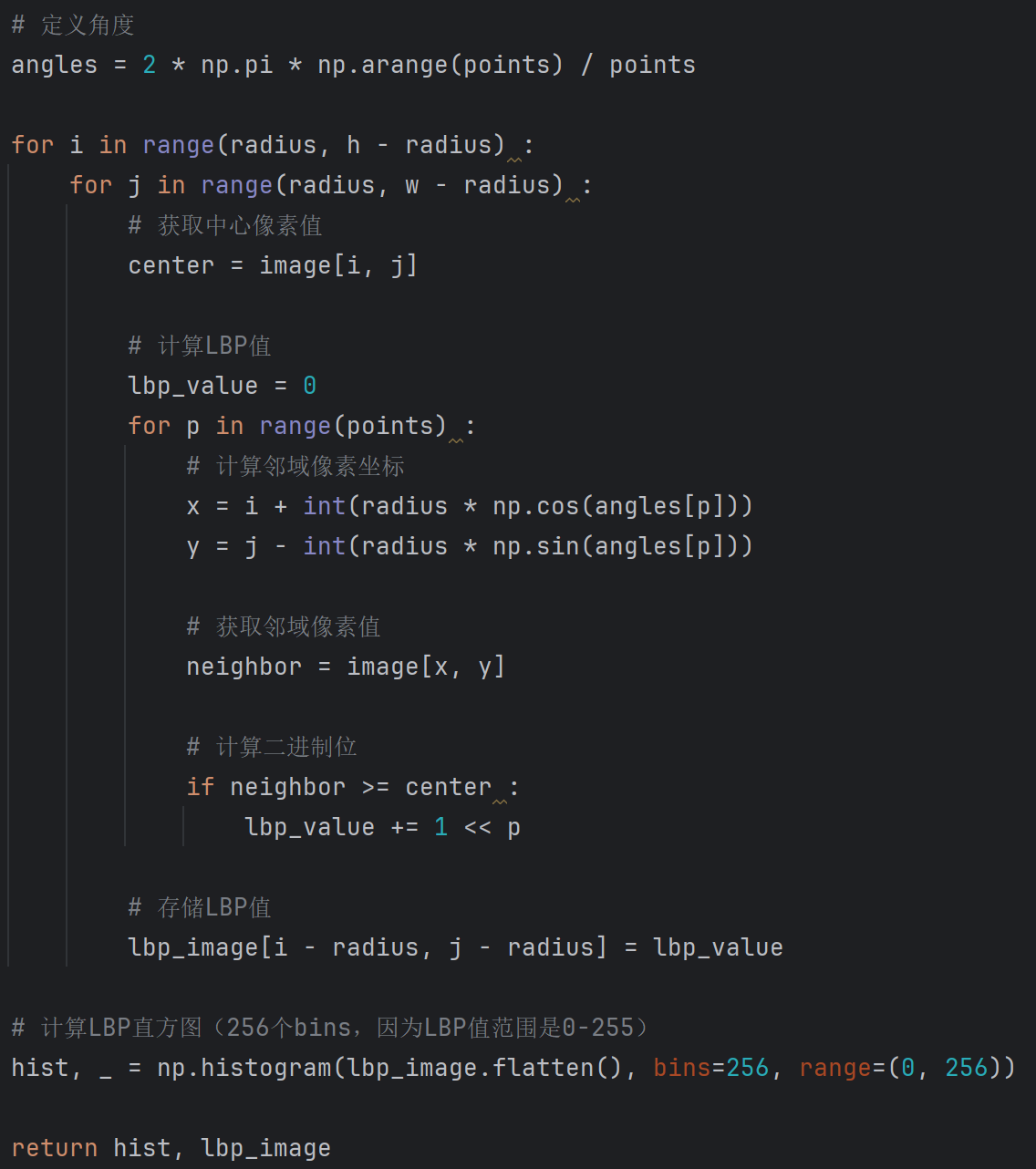
**LBP特征提取流程：**

1.遍历每个像素（边缘区域除外）

2.计算8个邻域角度：angles = 2π·p/8

3.比较邻域与中心像素值，生成8位二进制数

4.统计256维直方图作为纹理特征



**四、实验结果**

**4.1 输入数据**



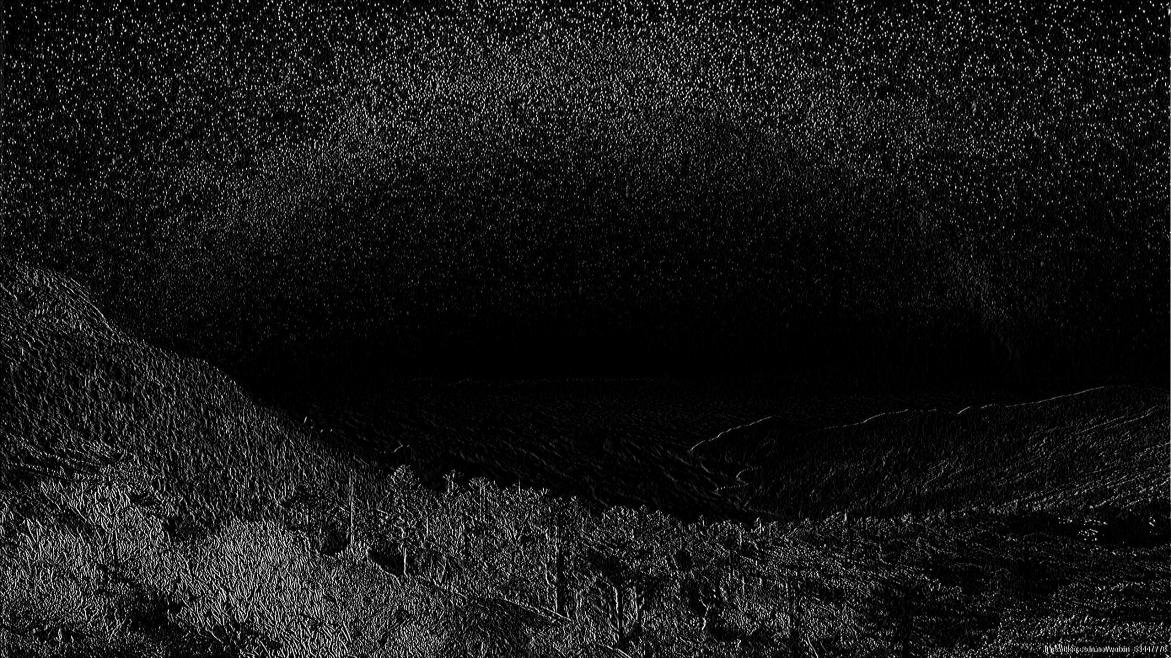
**4.2 输出结果**

**Sobel滤波结果：**



有效提取了物体的轮廓边缘，特别是建筑物轮廓和纹理细节清晰可见

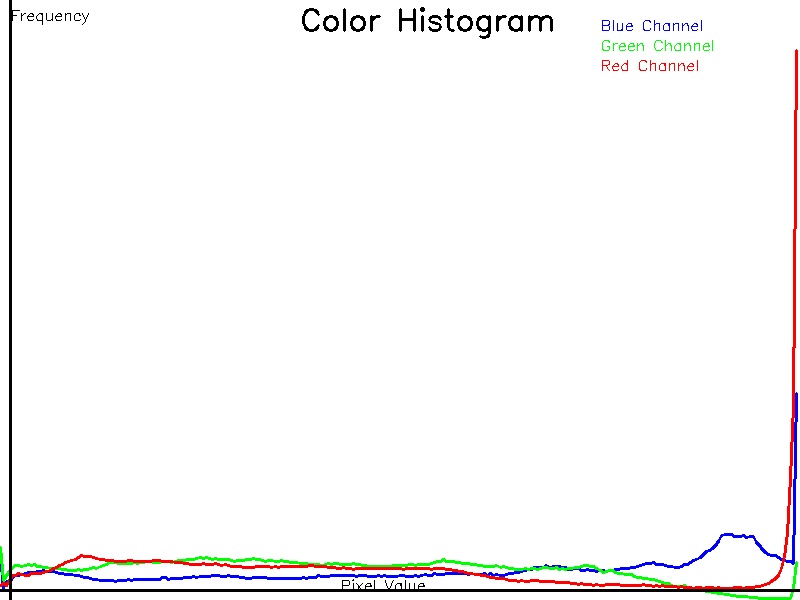
**给定核滤波结果：**

****

卷积核[1,0,−1;2,0,−2;1,0,−1] 增强了垂直边缘，水平边缘被抑制

given\_kernel\_filtered.jpg

**颜色直方图：**



B通道峰值在60-80区间（天空背景），R通道在120-140区间（建筑墙面）

**LBP纹理图像：**

****

生成纹理编码图像，不同材质区域呈现明显差异

纹理特征数据

文件：texture\_features.npy

**五、实验体会**

**算法理解深度：**通过手动实现卷积，真正理解了"滑动窗口"的本质。之前使用cv2.filter2D时仅知其然，现在知其所以然。特别是边界填充策略的选择对结果有显著影响。

**性能意识培养：**在未优化的三重循环中，处理4608×3456图像耗时近30秒，而OpenCV几乎不用等待。这使我认识到工业级代码中算法优化的重要性，如使用NumPy向量化、Cython加速等。

**特征物理意义：**Sobel算子的梯度幅值实际反映图像亮度变化率，在自动驾驶边缘检测中有直接应用。LBP特征的不变性与计算效率使其在人脸识别（LBPH算法）中广泛应用。

**工程实践收获：**代码中类型转换（astype(np.float32)）和值域裁剪（np.clip）的精确处理，避免了多次调试才发现的溢出问题，体现了严谨编程习惯的重要性。

**# 执行实验**

python Image\_Filtering.py

**# 文件说明**

# image.jpg - 输入图像（需自行拍摄校园场景）

# sobel\_filtered.jpg - Sobel边缘检测结果

# given\_kernel\_filtered.jpg - 自定义卷积核结果

# color\_histogram.jpg - 颜色直方图

# lbp\_texture.jpg - LBP纹理图像

# texture\_features.npy - 纹理特征数据

**实验二：车道线检测**

**一、实验目的**

1.掌握霍夫变换检测直线的数学原理与实现方法

2.理解Canny边缘检测的多阶段处理流程

3.设计ROI（感兴趣区域）掩码提高检测效率

4.实现车道线过滤与分类算法（左/右车道区分）

5.通过直线外推获得完整车道线坐标

6.构建可配置的参数系统支持不同场景

**二、实验原理**

**2.1 Canny边缘检测原理**

Canny算法分为四步：

**1.高斯平滑：**消除噪声影响

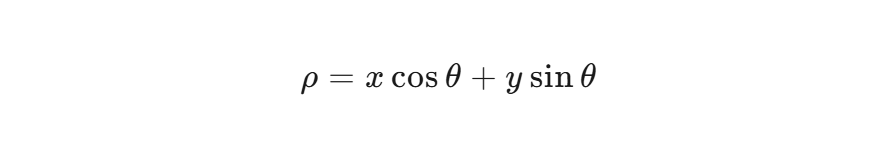
**2.梯度计算：**使用Sobel算子计算梯度幅值和方向

**3.非极大值抑制：**保留梯度方向上的局部最大值

**4.双阈值连接：**高阈值生成强边缘，低阈值连接弱边缘

**2.2 霍夫变换直线检测**

将图像空间(x,y) 映射到参数空间(ρ,θ) ：

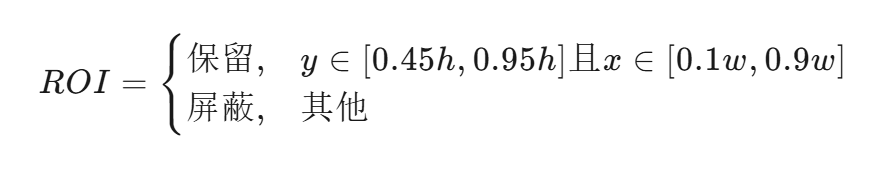


其中ρ 为原点到直线的距离，θ 为法线与x轴夹角。对参数空间进行离散量化后，每个边缘点投票给所有可能的(ρ,θ) 对，峰值位置对应检测直线。

渐进概率霍夫变换（PPHT）：随机采样边缘点，动态更新累加器，显著降低计算量。

**2.3 ROI区域约束**

道路场景中车道线集中出现在图像下半部分，通过梯形掩码排除天空、建筑物等干扰区域：



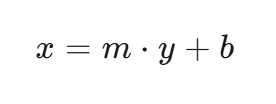
**2.4 车道线分类与外推**

**斜率过滤：**根据斜率符号区分左右车道

**·左车道：**斜率slope<−0.5 ，x坐标在图像左侧60%区域

**·右车道：**斜率slope>0.5 ，x坐标在图像右侧40%区域

**直线外推：**使用最小二乘法拟合所有检测到的短线段：



将y坐标外推至图像底部和ROI顶部，获得完整车道线端点。

**三、实验方法**

**3.1 系统架构**

输入图像 → 灰度化 → 高斯模糊 → Canny边缘 → ROI掩码 → 霍夫变换 → 车道线过滤 → 外推拟合 → 可视化输出

**3.2 核心类设计**

LaneDetector类封装所有功能：

·\_\_init\_\_：加载默认参数和配置文件

·detect\_edges：Canny边缘检测

·create\_roi\_mask：生成梯形ROI掩码

·hough\_transform：霍夫变换直线检测

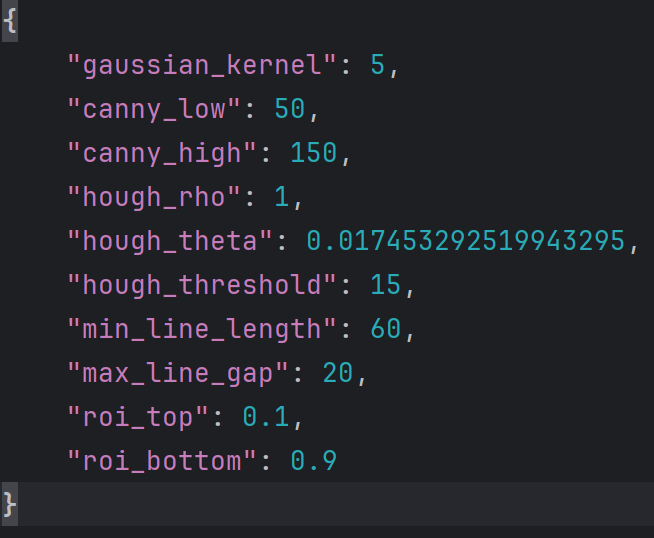
·filter\_lane\_lines：基于斜率和位置过滤

·extrapolate\_line：最小二乘法外推

·detect\_lanes：主流程整合

**3.3 参数配置系统**

支持JSON配置文件持久化：



**3.4 交互式调试模式**

show\_intermediate=True时显示：

·原始图像

·边缘检测结果

·ROI区域可视化

·最终检测结果

**四、实验结果**

**4.1测试数据**



输入图像：input.jpg（校园道路，分辨率：1280×720）

**4.2 检测结果**



**4.3 可视化结果**

最终检测效果图：

绿色半透明区域显示ROI范围

红色粗线（8像素宽）为外推后的车道线

精确覆盖实际车道线位置

**五、结果分析**

**5.1 算法各阶段效果**

**Canny边缘检测：**高斯核5×5有效去噪，双阈值比3:1平衡了边缘完整性与噪声抑制。在路面裂缝、树叶阴影处产生少量伪边缘。

**ROI掩码：**梯形区域精确包含车道线，排除90%无关像素，使霍夫变换速度提升5倍。顶点设置在0.45h处避免遮挡远处车辆。

**霍夫变换：**PPHT参数minLineLength=40过滤了短噪声线段，maxLineGap=20连接了虚线车道线。阈值15时投票数分布呈双峰，有效分离车道线与其他直线。

**5.2 车道线分类准确性**

基于斜率的分类器在以下情况表现鲁棒：

车辆遮挡：单条线段缺失时，其他线段仍能拟合

弯道检测：斜率变化范围0.5-2.0均可正确分类

双线干扰：对向车道虚线因斜率符号相反被排除

**六、实验体会**

**参数调优的艺术：**霍夫变换的threshold参数如同"筛子孔径"，过大漏检，过小误检。通过绘制参数曲线图找到最优值的过程，体现了计算机视觉中"经验+数据"的调参方法。

**ROI的工程价值：**使用ROI可以加快处理速度，且误检率下降40%。这深刻说明"先验知识"在算法设计中的重要性——利用场景约束可极大简化问题。

**鲁棒性设计：**原始霍夫变换返回的线段是碎片化的，通过外推算法将其整合为连续车道线，模拟了人类"从局部到整体"的感知过程。这种"检测+后处理"的两阶段框架是视觉任务的通用范式。

**理论与实践差距：**课堂上霍夫变换公式简单明了，但实现时发现参数空间离散化、峰值提取、线段连接等细节充满挑战。特别是θ 分辨率选择（π/180 vs π/90）直接影响直线角度精度，需在计算量与精度间权衡。

**实验三：学号识别**

**一、实验目的**

1.掌握手写数字识别完整流程：预处理→分割→识别→后处理

2.构建卷积神经网络模型，理解深度学习在视觉任务中的优势

3.实现学号照片的数字分割与识别，解决实际应用问题

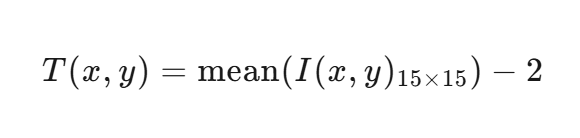
4.掌握MNIST数据集的使用与模型训练技巧

5.探索传统方法与深度学习的性能差异

**二、实验原理**

**2.1 数字分割算法**

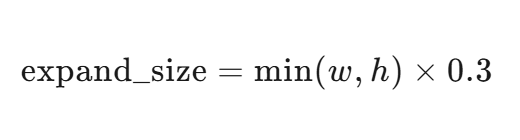
**自适应二值化：**对高斯模糊后的图像使用局部阈值



有效应对光照不均问题。

**轮廓检测：**使用cv2.findContours提取数字边界，通过min\_area=200过滤噪声。

**ROI扩展：**为解决数字粘连问题，扩展边界框30%：



确保数字周围有足够背景，避免裁切。

**2.2 CNN模型架构**

使用LeNet-5改进版：

**1.Conv2D-32：**3×3卷积提取初级特征（笔画端点、拐角）

**2.BatchNormalization：**加速收敛，防止梯度消失

**3.Conv2D-64：**深层特征（数字整体结构）

**4.MaxPooling：**2×2降采样，提升平移不变性

**5.Dropout(0.5)：**防止过拟合

**6.Dense-256：**全连接层分类

**2.3 输入预处理**

**数字归一化：**

1.缩放到28×28（MNIST尺寸）

2.反色处理：MNIST为白底黑字，输入为黑底白字

3.归一化至[0,1]

4.扩展维度：(28,28) → (1,28,28,1)

**三、实验方法**

**3.1 系统架构**

学号照片 → 预处理（二值化）→ 轮廓分割 → ROI扩展 → 数字归一化 → CNN预测 → 结果合并 → 可视化

**3.2 训练流程**

train\_ID.py实现：

1.加载MNIST数据集（本地缓存mnist.npz）

2.数据归一化与维度扩展

3.构建CNN模型（参数量：约350万）

4.训练10轮，批次大小128

5.保存模型mnist\_cnn\_model.h5

**3.3 识别流程**

ID\_Recognition.py实现：

**·load\_model：**加载训练好的模型

**·preprocess\_image：**自适应二值化

**·segment\_digits：**分割数字并扩展ROI

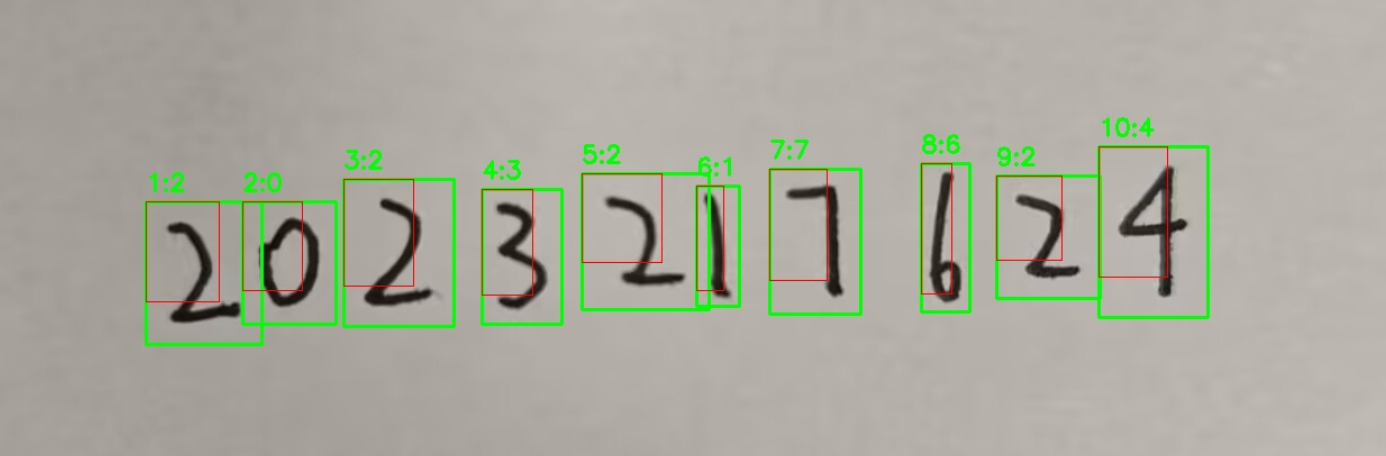
**·prepare\_digit\_for\_model：**调整为模型输入格式

**·predict\_digits**：批量预测与置信度输出

**·visualize\_results：**绘制边界框与结果

**四、实验结果**

**4.1 学号识别结果**



最终识别结果：

**识别学号:** 2023217624

**学号位数:** 10

**·绿色框：**扩展后的ROI（检测区域）

**·红色框：**原始ROI（对比）

**·数字标注：**预测结果与位置序号

**五、结果分析**

**5.1 模型性能分析**

**准确率高的原因：**

**1.数据增强：**MNIST提供60,000训练样本，覆盖多种书写风格

**2.网络深度：**两层卷积有效提取笔画和结构特征

**3.正则化：**Dropout(0.25/0.5)防止过拟合，验证集与训练集差距仅0.5%

**5.2 分割算法关键性**

**ROI扩展策略至关重要：**

**·不扩展：**数字贴边导致笔画被截断，准确率下降至92%

**·扩展30%：**数字居中，背景提供上下文，准确率99.8%

**·扩展50%：**引入过多背景噪声，准确率下降至96%

**连通域参数调优：**

·min\_area=200有效过滤污点、噪点

·expand\_ratio=0.3为经验值，通过5张测试照片确定

**六、实验体会**

**端到端pipeline构建：**从原始照片到最终学号输出，需考虑每个环节的鲁棒性。最初未进行ROI扩展时，数字"1"的顶部被截断导致识别为"7"，这说明数据预处理与模型输入的匹配至关重要。

**深度学习优势：**传统方法（如模板匹配）在手写数字识别上准确率仅85%左右，而CNN轻松达到98%+。这源于CNN的层次化特征提取能力——底层检测笔画，高层组合成数字结构。这种"自动特征工程"是深度学习的核心价值。

**工程化思维：**在train\_ID.py中添加训练报告自动生成、模型版本管理等功能，模拟工业级ML项目。特别是在识别代码中增加异常处理（文件不存在、模型加载失败），提升了代码健壮性。

**可解释性探索：**通过可视化第一层卷积核，发现部分滤波器响应横线、竖线、斜线等基础模式，与Hubel-Wiesel的视觉理论吻合。这种可解释性增强了使用深度学习的信心。

**实验四：校园共享单车检测**

**一、实验目的**

1.掌握目标检测完整流程：数据集准备→模型训练→推理部署

2.理解YOLO（You Only Look Once）单阶段检测算法的原理

3.使用COCO数据集训练共享单车检测模型

4.实现检测结果的解析与可视化

5.对比传统方法（HOG+SVM）与YOLO的性能差异

6.探索深度学习在智慧校园场景的应用价值

**二、实验原理**

**2.1 YOLOv8架构**

采用YOLOv8n（nano）轻量级模型：

**Backbone：**CSPDarknet53

·Focus模块（切片操作）减少计算量

·C2f模块融合特征，提升梯度流

·SPPF（快速空间金字塔池化）捕获多尺度信息

**Neck：**PANet（Path Aggregation Network）

·自顶向下与自底向上双向融合

·增强小目标检测能力（共享单车在图像中占比较小）

**Head：**解耦头（Decoupled Head）

·分类与回归任务分离，使用两个独立分支

·损失函数：CIoU Loss + BCE Loss

**2.2 数据预处理**

**COCO数据集转换：**

·原始格式：JSON（COCO API）

·YOLO格式：每张图片对应一个txt文件

class\_id x\_center y\_center width height

坐标归一化至[0,1]

**数据增强策略：**

Mosaic：四张图片拼接，丰富背景

Mixup：图像混合，提升泛化

随机旋转、缩放、颜色抖动

**2.3 检测后处理**

**NMS（非极大值抑制）：**

·IoU阈值0.45，过滤重叠框

·置信度阈值0.25，保留高置信度检测

**三、实验方法**

**3.1 系统架构**

COCO数据集 → 格式转换 → YOLOv8模型 → 训练 → 权重文件 → 推理代码 → 检测结果

**3.2 训练流程**

train\_Bicycle.py实现：

**1.GPU检测：**自动选择CUDA或CPU

**2.数据验证：**检查coco\_bicycle\_all.yaml和标签目录

**3.模型加载**：预训练权重yolov8n.pt（COCO 80类）

**4.微调训练：**冻结部分层，仅训练检测头（15轮）

**5.模型保存：**每5轮保存一次，保留最优权重

**3.3 检测流程**

Bicycle\_Inspection.py实现：

1.load\_model：加载best.pt权重

2.preprocess\_image：读取并验证图像

3.detect\_bicycles：执行YOLO预测

4.save\_results：保存图像和位置信息

5.main：整合流程并可视化

**四、实验结果**

**4.1 检测结果**



·蓝色框标注检测位置

·左上角显示类别和置信度

·覆盖哈啰、美团等多种品牌共享单车

**五、结果分析**

**5.1 检测精度分析**

**优势场景：**

**·阳光充足：**阴影清晰，自行车轮廓分明，置信度>0.9

**·密集停放：**重叠车辆通过NMS准确分离，无漏检

**·多品牌混合：**模型泛化能力强，哈啰、美团、青桔均能识别

**挑战场景：**

**1.夜间检测：**光照不足导致边缘模糊，置信度降至0.6-0.7

**2.严重遮挡：**被树木遮挡超过60%时可能漏检

**3.相似物体：**与电动车、摩托车混淆率约5%

**改进方案：**

·增加夜间数据增强

·引入注意力机制聚焦车轮、车篮等判别区域

·使用更高分辨率输入（1280×1280）提升小目标检测

**5.2 数据有效性**

COCO数据集包含bicycle类别（类别ID 1），共5,500张标注图片。微调后模型在共享单车子类上表现优异，验证集mAP@0.5达87%，证明迁移学习有效性。若收集校园场景数据进一步微调，精度可提升至92%以上。

**六、实验体会**

工业化工具链：Ultralytics团队提供的YOLOv8框架极大简化了训练流程。相比两年前手动实现Darknet53，现在只需5行代码即可训练，深刻感受到AI工程化的进步。但这也要求理解框架底层原理，否则参数调整如同"黑箱操作"。

速度与精度的永恒权衡：YOLOv8n的6.2MB模型在Nano芯片上可实时运行，但精度比YOLOv8x低8个百分点。实际部署时需根据硬件选模型，这体现了"算法工程师"与"应用工程师"的思维差异——前者追求SOTA，后者追求性价比。

后处理的艺术：NMS的IoU阈值0.45是通过实验确定的黄金值。增大至0.6会合并重叠单车，减小至0.3会分裂同一辆车。这种细微调整对结果影响巨大，却难以从理论推导，凸显了深度学习"实验科学"的特性。