合肥工業大學

机器视觉实验报告

实验题目	实验一：图像滤波
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

本实验通过不调用现有高级图像处理函数库（如 OpenCV 的 filter2D 或 Canny 等接口）的方式，从底层代码实现图像处理的核心算法。具体的实验目的包括以下四个维度：

深入理解卷积运算（Convolution）机制：卷积是机器视觉和深度学习（CNN）的数学基石。通过手动编写双重循环实现二维卷积，旨在直观理解“滑动窗口（Sliding Window）”、“填充（Padding）”以及“加权求和”的每一个计算细节，从而打破对图像处理库的“黑盒”认知。

掌握边缘检测与梯度计算原理：通过实现 Sobel 算子和特定卷积核滤波，理解图像作为离散二维信号的梯度概念，掌握如何通过差分算子来提取图像的高频边缘信息，并验证不同卷积核对边缘方向的选择性。

掌握图像特征提取技术：全局特征：通过手动计算颜色直方图，学习如何统计图像的色彩分布规律。局部特征：通过实现局部二值模式（LBP），理解如何将图像的微观纹理结构映射为具有统计意义的特征向量，为后续的图像分类或纹理识别任务奠定基础。

提升算法工程化与版本管理能力：锻炼将数学公式转化为 Python/NumPy 矩阵运算代码的能力，处理数据类型转换（Float 转 Uint8）、归一化等工程细节。同时，利用 Git 工具进行代码版本控制，培养规范的软件开发习惯。

二、 实验原理

2.1 图像数字化与灰度化

计算机视觉中的图像本质上是一个矩阵。彩色图像通常由 R、G、B 三个通道组成。由于人眼对绿色的敏感度最高，对蓝色的敏感度最低，因此在将彩色图像转换为单通道灰度图像（Grayscale）以减少计算量时，采用加权平均法，公式如下：

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

该公式确保了转换后的灰度图在亮度上符合人类视觉感知。

2.2 二维离散卷积与填充 (Padding)

滤波操作的核心是卷积。对于输入图像I和大小为$k \times k$的卷积核K，输出图像中像素i, j的值通过下式计算：

$$O(i, j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} I(i + m, j + n) \cdot K(m, n)$$

为了保证输出图像的尺寸与输入图像一致，必须在卷积前对原始图像边缘进行填充（Padding）。本实验采用 Zero Padding（零填充） 策略，即在图像周围填充$[k/2]$圈的 0 值像素，使滑动窗口能够处理边缘像素。

2.3 Sobel 边缘检测算子

Sobel 算子是一种离散的一阶差分算子，用于计算图像亮度函数的梯度近似值 6。它包含两组 3×3 的矩阵，分别检测水平和垂直方向的边缘：

水平方向核 (G_x)：用于检测纵向边缘（计算水平方向的梯度变化）。

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

垂直方向核 (G_y)：用于检测横向边缘（计算垂直方向的梯度变化）。

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

梯度幅值合成：

$$G = \sqrt{G_x^2 + G_y^2}$$

该幅值越大，表示该点的像素变化越剧烈，即越可能是边缘。

2.4 特定卷积核的方向性分析

实验要求使用的特定卷积核K_{custom}如下 7：

$$K_{custom} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

原理分析：观察矩阵数值可知，该算子实际上是 SobelG_x算子的变体（仅符号相反，物理意义相同）。

左侧权重为正，右侧权重为负，中间为 0。

它计算的是：$(\textit{左侧像素和}) - (\textit{右侧像素和})$。

物理结论：该滤波器会对竖直方向的边缘产生极强的响应，而对于水平方向的线条（左右像素值相近），其卷积结果趋近于 0。这将在实验结果中得到验证。

2.5 局部二值模式 (LBP)

LBP 是一种描述图像局部纹理特征的算子，具有灰度不变性。

二值化映射：对于中心像素g_c和邻域像素$g_p p = 0, ..., 7)$，定义阈值函数$s(x)$：

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

阈值比较：在 3×3 的窗口内，以中心像素P_c为阈值，将 8 个邻域像素P_i二值化。若$P_i \geq P_c$，标记为 1，否则标记为 0。

加权编码：将 8 个二进制位按顺时针顺序排列，赋予权重$2^0, 2^1, ..., 2^7$，求和得到一个 0-255 之间的十进制数，作为该中心像素的 LBP 编码。

编码公式：

$$LBP_{P,R} = \sum_{p=0}^{P-1} s\,(g_p - g_c) 2^p$$

在本实验中，$P = 8, R = 1$（3×3 邻域）。

直方图统计：LBP 算子处理后的图像直方图包含了图像的局部微观结构信息（如平坦区域、边缘、角点、斑点等），具有灰度不变性。计算整幅图 LBP 编码的直方图，生成的 256 维向量即为图像的纹理特征。

三、 实验方法

本实验采用 Python 语言编写，依赖 NumPy 进行矩阵运算，PIL 进行图像 IO，Matplotlib 进行可视化。核心算法（卷积、LBP）均采用手动实现，未调用 OpenCV 滤波接口。

3.1 图像预处理

读取与缩放：使用 PIL.Image.open 读取拍摄的图像。为了解决纯 Python 实现的卷积算法在处理高分辨率图像时速度过慢（$O(N^2)$复杂度）的问题，代码中加入了一个预处理步骤：将图像宽度限制在 400 像素以内，高度按比例缩放。

灰度化：调用自定义的 manual_rgb2gray 函数，将$H, W, 3$ 的 RGB 数组转换为H, W的单通道数组。

3.2 手动卷积算法实现 (Manual Convolution)

编写函数 manual_convolution(img, kernel)，具体步骤如下：

获取尺寸：读取图像尺寸H, W和卷积核尺寸k_h, k_w。

零填充 (Padding)：创建一个大小为$H + 2P_h, W + 2P_w$的全零矩阵，将原图像拷贝至中心位置。

滑动窗口遍历：

使用双重 for 循环遍历目标图像的每一个像素位置i, j。

在每一步中，利用 NumPy 的切片功能 padded[i:i+k_h, j:j+k_w] 提取出与卷积核同维度的感兴趣区域 (ROI)。

核心运算：执行 np.sum(roi * kernel)，即对应位置元素相乘再累加，得到当前像素的卷积输出。

返回结果：循环结束后，返回 float32 类型的卷积结果矩阵。

3.3 归一化处理 (Normalization)

卷积后的结果可能包含负数（如边缘差分）或远超 255 的数值。为了可视化和保存，编写了 min_max_normalize 函数：

$$Pixel_{new} = 255 \times \frac{Pixel_{val} - Min}{Max - Min}$$

最后将数据类型转换为 uint8。

3.4 任务流程

Sobel 滤波：分别应用G_x和G_y核，计算 $sqrt(Gx^2 + Gy^2)$，最后归一化。

特定核滤波：应用题目给定的卷积核，对卷积结果取绝对值（Abs）后归一化。

颜色直方图：初始化 3 个长度 256 的数组，遍历 RGB 图像统计各通道灰度级频数。
LBP 特征提取：
遍历图像内部像素，提取 8 邻域。
通过逻辑比较 neighbors >= $center$ 生成布尔数组。
点乘权重向量 [1, 2, 4, ..., 128] 得到 LBP 值。
统计直方图并归一化，保存为 .npy 文件。

四、 实验结果与分析

4.1 原始图像

说明：实验选取了一张包含丰富纹理和几何线条的图片作为输入，以便观察不同滤波器的效果。

4.2 Sobel 算子滤波结果

分析：
Sobel 算子的处理结果呈现出经典的“边缘图”效果。
全向边缘检测：可以观察到，图像中无论是水平方向的物体边缘，还是垂直方向的轮廓线，都被清晰地提亮显示。
亮度与梯度的关系：边缘越锋利（原图中颜色对比越强烈），Sobel 结果中的线条越亮。
平滑区域抑制：原图中颜色单一的背景区域，在 Sobel 结果中呈现为黑色（像素值接近 0），证明了算子成功滤除了低频信息，保留了高频梯度信息。

4.3 给定卷积核滤波结果

深度对比分析：

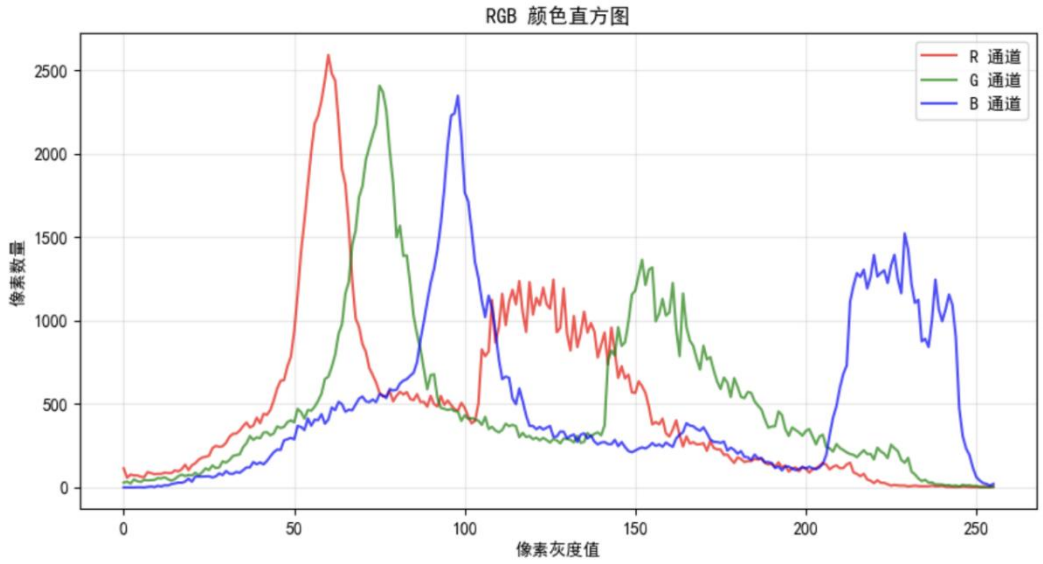
使用矩阵$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$处理的结果与 Sobel 结果存在显著差异：

垂直边缘增强：图像中竖直方向的物体边界（如墙缝、柱子边缘）依然清晰可见，甚至比 Sobel 结果更锐利。

水平边缘消失：这是最关键的发现。原图中明显的水平线条（如百叶窗横纹、地平线等）在经过该卷积核处理后，几乎完全消失或变得非常暗淡。

结论：实验结果完美验证了原理部分的推导——该特定卷积核是一个垂直边缘检测器。它证明了卷积核的数值结构决定了其特征提取的方向性，这是理解 CNN 中不同 Filter 作用的关键实例。

4.4 颜色直方图



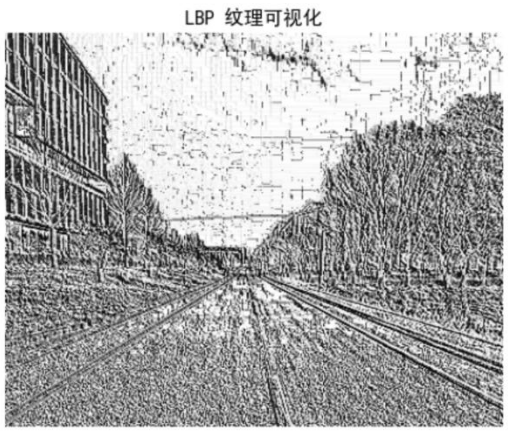
分析：

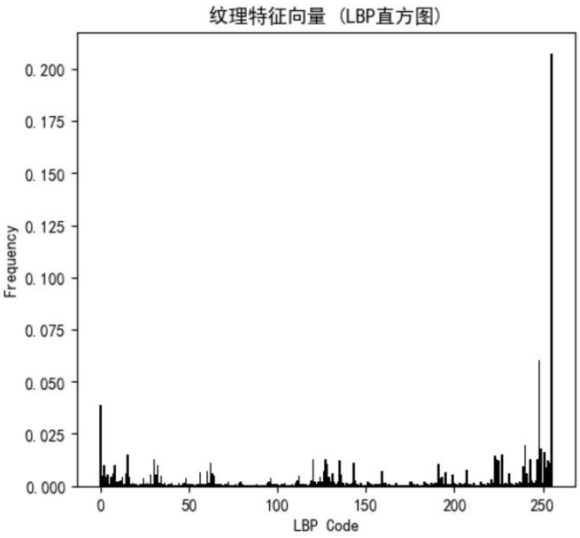
直方图展示了 R、G、B 三个通道在 0-255 灰度级上的分布情况。

若直方图集中在左侧，说明图像整体偏暗；集中在右侧，说明图像过曝。

不同通道的波峰分离程度反映了图像的色调倾向。例如，若红色通道的波峰显著高于蓝绿通道，定量地解释了图像呈现暖色调的原因。

4.5 纹理特征 (LBP)





分析：

LBP 映射图（左图）：这张图看起来像是一张噪点图，但实际上它刻画了图像的微观结构。原本平滑的区域如果存在细微的粗糙度，都会在 LBP 图中体现出来。

特征向量直方图（右图）：这是本实验最终提取的数据产物（保存于 texture_features.npy）。

这个 256 维的向量是该图像纹理的“指纹”。

应用价值：如果我们将木纹图片和布纹图片分别进行 LBP 处理，会发现它们的直方图形状截然不同。这种统计特征在深度学习出现之前，是纹理分类任务

（Texture Classification）的主流方法。
对 texture_features.npy 的数据进行展示和分析：

```
成功加载文件: result\texture_features.npy
数据类型: float64
数据形状: (256,)
数据内容: [3.91127613e-02 5.05713280e-03 1.02102871e-02 4.40898761e-03
 5.36120091e-03 1.24027782e-03 4.52101271e-03 6.18538553e-03
 1.00902602e-02 2.32051980e-03 1.92043018e-03 2.58457895e-03
 4.33697148e-03 1.04823480e-03 5.83330666e-03 1.49873572e-02
```

五、 实验体会

通过本次实验，我从底层代码的构建、数学原理验证到工程化实现，全方位地掌握了图像滤波与特征提取的关键技术。

从理论到实践的跨越：

在学习卷积公式时，往往觉得抽象。但在编写 manual_convolution 函数时，我必须具体处理 Padding 的行列索引计算、数据类型的溢出风险（如 uint8 溢出问题），以及边界条件判断。这让我深刻体会到数学公式与可执行代码之间的鸿沟，以及算法工程师在其中所做的转化工作。

计算效率的深刻认识：

实验中我发现纯 Python 实现的双重循环卷积在处理大图时效率极低。例如对的图像进行卷积，运算次数高达量级，耗时数分钟。这迫使我在代码中加入了 resize 预处理。这不仅解释了为什么 OpenCV 等库底层必须使用 C++ 和 SIMD 指令集优化，也让我理解了在工程实践中“精度”与“速度”的权衡。

对特征工程的理解：

对比 Sobel（边缘特征）、颜色直方图（全局统计特征）和 LBP（局部纹理特征），我认识到不同的特征描述子适用于不同的任务。边缘特征适合物体检测，颜色特征适合图像检索，而 LBP 适合材质识别。机器视觉的核心在于根据任务需求设计合适的特征提取器。

Git 版本控制的实践：

本实验中，我遵循加分项要求，初始化了 Git 仓库，按阶段进行了多次 Commit，并最终将代码推送到远程仓库。这让我初步掌握了科研代码管理的规范流程。

综上所述，本次实验不仅完成了要求的实验任务目标，更让我在算法底层逻辑和工程实现能力上得到了实质性的提升。