实验一：图像滤波

林圳 2023217534 智科 23-1 班

一、实验目的

本实验通过手动实现图像处理领域中的基础算法,帮助我们深入理解数字图像处理的核心原理,掌握卷积运算的基本机制,并学会从图像中提取有意义的特征信息。具体包括：

(1) **理解并掌握卷积运算的基本原理**：卷积是图像处理中最基础也是最重要的运算之一,广泛应用于滤波、边缘检测、特征提取等多个领域。通过手动实现 2D 卷积操作,深入理解卷积核与图像之间的相互作用机制。

(2) **掌握 Sobel 边缘检测算法**：Sobel 算子是基于一阶导数的经典边缘检测算法,通过实验理解其工作原理、梯度计算方式以及如何通过 Gx 和 Gy 两个方向的综合来检测图像中的边缘信息。

(3) **学习并实现自定义滤波核**：在理解标准滤波器的基础上,学会如何应用自定义卷积核进行特定的图像处理任务,培养对卷积核设计的理解能力。

(4) **掌握颜色直方图的统计与可视化方法**：颜色直方图是描述图像全局颜色分布特征的重要工具,通过手写实现理解其统计原理和可视化方法。

(5) **深入理解灰度共生矩阵纹理特征提取**：GLCM 是一种描述图像纹理特征的重要方法,通过实验理解计算原理以及如何从 GLCM 中提取对比度、能量、同质性、熵等纹理特征。

二、实验原理

2.1 图像的数字表示与基本操作

2.1.1 图像的数据结构

数字图像本质上是一个二维数组,对于彩色图像而言,通常采用 RGB 色彩空间,由三个颜色通道(红、绿、蓝)组成。每个通道对应一个二维矩阵,矩阵中的每个元素代表该像素点的亮度值,取值范围通常为 0-255 的整数。在本实验中,使用 NumPy 数组来存储和处理图像数据,RGB 图像的维度为 H×W×3,其中 H 表示图像高度,W 表示图像宽度,3 表示 RGB 三通道。

2.1.2 RGB 到灰度的转换

灰度图像是单通道图像,每个像素点仅用一个数值表示亮度。将 RGB 图像转换为灰度图像需要考虑人眼对不同颜色亮度的感知差异。本实验采用的心理学灰度转换公式为：

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

该公式的权重来源于人眼对不同光谱的敏感度特性,绿色通道权重最大,红色次之,蓝色最小。这种转换方式能够较好地保持人眼对图像亮度的感知效果,是图像处理领域广泛采用的标准方法。

2.1.3 零填充(Zero Padding)

在进行卷积操作时,由于卷积核会遍历图像的每一个像素,位于图像边界的像素点无法形成完整的卷积窗口,这会导致输出图像尺寸减小。为了保持输出图像与输入图像尺寸一致,需要在图像的四周填充像素值。零填充是一种常用的填充方法,即在图像四周补零,使卷积核能够在边界位置也能完成完整的卷积运算。本实验通过 zeropad 函数实现了这个功能,填充的像素层数取决于卷积核的大小,计算公式为：pad = kernelsize // 2。

2.2 卷积运算原理

2.2.1 卷积的数学定义

卷积是信号处理中的基本运算,在图像处理中,2D 卷积的数学定义为：

$$(f * g)[i, j] = \sum\sum f[m, n] \times g[i-m, j-n]$$

其中 f 表示输入图像,g 表示卷积核,符号*表示卷积运算。在图像处理的实际应用中,通常采用相关运算而非严格的卷积运算,两者的区别在于卷积核是否需要旋转。由于旋转操作在实际实现中可以通过卷积核预先翻转来实现,因此在图像处理领域,相关运算也被统称为卷积运算。

2.2.2 离散 2D 卷积的实现

本实验通过手动实现了 2D 卷积运算,具体过程如下:

(1) 首先对输入图像进行零填充,确保输出图像与输入图像尺寸一致

(2) 然后通过四重循环遍历图像的每一个像素点和卷积核的每一个元素

(3) 对于图像的每一个位置(i,j),计算卷积核与该位置对应图像窗口的乘积之和

(4) 将计算结果赋给输出图像的对应位置

卷积运算的时间复杂度为 O(H×W×K×K),其中 H 和 W 是图像的尺寸,K 是卷积核的尺寸。对于较大的图像和较大的卷积核,卷积运算会消耗较多的计算资源。

2.2.3 卷积核的设计与作用

卷积核(也称滤波器)是卷积运算中的核心参数,其大小和权值决定了卷积运算的效果。卷积核的尺寸通常为奇数(如 3×3、5×5、7×7),这样便于确定中心位置。不同的卷积核可以实现不同的图像处理效果:

(1) **低通滤波核**:如均值滤波核,用于图像平滑和降噪

(2) **高通滤波核**:如拉普拉斯核,用于边缘检测和锐化

(3) **方向性滤波核**:如 Sobel 核,用于检测特定方向的边缘

2.3 Sobel 边缘检测原理

2.3.1 边缘检测的基本概念

边缘是图像中亮度发生剧烈变化的位置,是图像的重要特征之一。边缘检测的目标是定位这些亮度突变的位置,为后续的图像分析和理解提供基础信息。边缘检测算法主要分为基于一阶导数和基于二阶导数的方法,Sobel 算子属于基于一阶导数的方法。

2.3.2 Sobel 算子的数学原理

Sobel 算子利用图像的一阶导数来检测边缘,通过计算图像在水平和垂直两个方向的梯度近似值来识别边缘。Sobel 算子的核心思想是在每个像素点周围选取一个 3×3 的窗口,分别应用两个方向的卷积核来计算梯度近似值。

水平方向的 Sobel 卷积核 Gx 用于检测垂直方向的边缘:

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

垂直方向的 Sobel 卷积核 Gy 用于检测水平方向的边缘:

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

2.3.3 梯度计算与边缘幅值

对于图像中的每个像素点,分别使用 Gx 和 Gy 进行卷积运算,得到该点在两个方向的梯度分量 Gx 和 Gy。然后综合这两个方向的梯度,计算梯度幅值:

$$magnitude = \sqrt{(Gx^2 + Gy^2)}$$

梯度幅值反映了该位置的边缘强度,幅值越大表示边缘越明显。在实际应用中,有时也会使用简化的梯度计算公式来减少计算量,如 Gx+Gy 的绝对值之和,但这种简化会损失一定的精度。

2.3.4 Sobel 算子的特点

Sobel 算子具有以下特点：

(1) 引入了加权机制,对中心位置的像素赋予更高的权重,增强了抗噪能力

(2) 实现简单,计算效率较高,适合实时应用

(3) 对噪声具有一定的抑制作用,相比简单的差分算子更加稳健

(4) 能够同时检测水平和垂直方向的边缘,对角线方向的边缘也能被较好地识别

2.4 颜色直方图原理

2.4.1 直方图的基本概念

直方图是一种统计图形,用于表示数据分布情况。在图像处理中,颜色直方图描述了图像中各个颜色级别出现的频次。对于 RGB 彩色图像,可以分别统计 R、G、B 三个通道的直方图,每个通道有 256 个灰度级别(0-255)。

2.4.2 颜色直方图的统计方法

颜色直方图的统计过程为：遍历图像的每一个像素点,统计每个颜色级别的出现次数。对于彩色图像,分别统计 R、G、B 三个通道的直方图。直方图的形状能够反映图像的颜色分布特征：

(1) 直方图分布在较暗区域表示图像整体偏暗

(2) 直方图分布在较亮区域表示图像整体偏亮

(3) 直方图分布均匀表示图像的对比度较好

(4) 直方图集中在某些区域表示图像的颜色分布不均匀

2.4.3 直方图的应用

颜色直方图在图像处理中有广泛的应用,包括：

(1) 图像质量评估：通过分析直方图分布判断图像的曝光情况

(2) 图像检索：直方图可以作为图像的全局特征用于图像检索

(3) 图像增强：通过直方图均衡化提高图像对比度

(4) 图像分割：基于直方图的阈值分割算法

2.5 灰度共生矩阵(GLCM)纹理特征原理

2.5.1 纹理的概念

纹理是图像的重要视觉特征之一,描述了图像表面的结构信息和重复模式。与颜色特征不同,纹理特征关注的是像素之间的空间排列关系,反映图像的微观结构特征。常见的纹理包括平滑纹理、粗糙纹理、规则纹理、随机纹理等。

2.5.2 灰度共生矩阵的定义

灰度共生矩阵是描述图像纹理特征的重要方法,由 Haralick 在 1973 年提出。GLCM 统计图像中特定距离和方向上像素对共生的频次。设图像的灰度级数为 L,则在距离为 d、方向为 θ 的条件下,GLCM 是一个 L×L 的矩阵,元素 P(i,j)表示灰度级为 i 的像素与灰度级为 j 的像素在指定距离和方向上同时出现的概率。

2.5.3 GLCM 的参数设置

GLCM 的计算需要设置三个参数：

(1) **灰度级数**：为了减少计算量,通常需要对原始灰度图像进行量化处理,将 256 级灰度压缩到较小的级数(如 16 级)

(2) **距离**：指定像素对之间的距离,通常取值为 1

(3) **方向**：指定像素对的方向,常用方向包括 0°（水平)、45°（右下对角线)、90°（垂直)、135°（右上对角线)

在本实验中,灰度级数设置为 16,距离设置为 1,方向设置为 0°（水平向右)。

2.5.4 GLCM 纹理特征提取

从 GLCM 中可以提取多种纹理特征,本实验提取了五个经典特征：

(1) **对比度**：反映纹理的清晰程度和深浅程度

$$\text{对比度} = \sum (i-j)^2 \times P(i,j)$$

对比度越大,纹理越粗糙；对比度越小,纹理越平滑。

(2) **能量**：反映图像纹理的均匀程度和规律性

$$\text{能量} = \sum P(i,j)^2$$

能量越大,纹理越均匀规律；能量越小,纹理越混乱。

(3) **同质性**：反映图像纹理的局部平滑程度

$$\text{同质性} = \sum P(i,j) / (1 + |i-j|)$$

同质性越大,纹理越平滑；同质性越小,纹理变化越剧烈。

(4) **熵**：反映图像纹理的随机性和信息量

$$\text{熵} = -\sum P(i,j) \times \log_2 (P(i,j))$$

熵越大,纹理越复杂随机；熵越小,纹理越简单规则。

(5) **相关性**：反映 GLCM 在行或列方向的相似程度

$$\text{相关性} = \sum [(i-\mu i)(j-\mu j)P(i,j)] / (\sigma i \times \sigma j)$$

其中 μi、μj 分别为 GLCM 行、列方向的均值,σi、σj 分别为标准差。相关性越大,纹理的方向性越强。

这些纹理特征从不同角度描述了图像的纹理特性,在图像分类、纹理分割、医学影像分析等领域有广泛应用。

三、实验方法

3.1 实验环境与工具

3.1.1 硬件环境

本实验在远程服务器环境下进行（我在智谱实习期间给的服务器),具体配置如下：

(1) GPU：8 卡 A100

(2) 内存：80GB

(3) 存储：200T

3.1.2 软件环境

实验使用 Python 编程语言,主要依赖的第三方库包括：

(1) NumPy：用于高效的数值计算和数组操作,是本实验的核心工具

(2) PIL(Pillow)：用于图像的加载和保存

(3) Matplotlib：用于直方图和 GLCM 热力图的可视化

3.2 实验流程设计

本实验采用模块化设计,将完整的图像处理流程分解为多个独立的功能模块,每个模块负责特定的处理任务。实验流程主要包括以下步骤：

3.2.1 模块一：图像加载与预处理

(1) **图像加载**：使用 PIL 库的 Image.open()函数读取图像文件,通过 convert("RGB")强制转换为标准的 RGB 格式,确保后续处理的兼容性

(2) **灰度转换**：手动实现 RGB 到灰度的转换,采用心理学灰度公式 Gray = 0.299×R + 0.587×G + 0.114×B,为边缘检测和纹理特征提取做准备

(3) **数据类型处理**：将图像数据从 uint8 类型转换为 float32 类型,以便进行卷积运算等需要精确计算的操作

3.2.2 模块二：卷积运算实现

本模块实现了通用的 2D 卷积函数,是 Sobel 滤波和自定义滤波的基础。

(1) **零填充**：实现 zero_pad()函数,在图像四周填充指定层数的零值,防止卷积后图像尺寸减小

(2) **卷积核验证**：检查卷积核是否为奇数方阵,确保具有中心位置

(3) **卷积运算**：通过四重循环实现完整的 2D 卷积过程：

① 外两层循环遍历图像的每个像素位置

② 内两层循环遍历卷积核的每个元素

③ 计算卷积核权值与对应图像像素的乘积之和

(4) **输出处理**：返回与输入图像尺寸相同的卷积结果

3.2.3 模块三：Sobel 边缘检测

(1) 定义 Sobel 算子：创建 3×3 的 Gx 和 Gy 卷积核,分别用于检测垂直和水平方向的边缘

(2) 卷积计算：调用通用的卷积函数,分别用 Gx 和 Gy 对灰度图像进行卷积

(3) 梯度幅值计算：根据公式 magnitude = $\sqrt{(Gx^2 + Gy^2)}$综合两个方向的梯度

(4) 结果归一化：将梯度幅值归一化到 0-255 范围的 uint8 类型,便于保存和可视化

3.2.4 模块四：自定义卷积核滤波

(1) **定义自定义卷积核**：创建实验要求的特定卷积核[[1,0,-1],[2,0,-2],[1,0,-1]],该核类似于 Sobel Gx 核

(2) **卷积处理**：使用通用卷积函数对灰度图像进行滤波

(3) **结果归一化**：与 Sobel 滤波一样,将结果转换为 uint8 类型保存

3.2.5 模块五：颜色直方图统计

(1) 初始化直方图数组：创建一个 3×256 数组,分别存储 R、G、B 三个通道的直方图数据

(2) 手动统计：通过双层循环遍历图像的每个像素,手动统计每个通道每个灰度级别的出现频次

(3) 可视化：使用 Matplotlib 绘制三通道的颜色直方图曲线图,便于观察和分析

3.2.6 模块六：GLCM 纹理特征提取

(1) **灰度量化**：将 0-255 的灰度级压缩到 16 级,减小 GLCM 矩阵的尺寸和计算复杂度

(2) **GLCM 计算**：遍历图像,统计指定方向和距离上像素对共生的频次

(3) **概率归一化**：将 GLCM 从频次矩阵转换为概率矩阵,使所有元素之和为 1

(4) **特征提取**：从归一化的 GLCM 中计算对比度、能量、同质性、熵、相关性五个纹理特征

(5) **热力图可视化**：使用 Matplotlib 将 GLCM 绘制为热力图,直观展示纹理共生的分布

3.2.7 模块七：结果保存与可视化

(1) **图像保存**：将处理后的边缘检测结果、自定义滤波结果、Gx/Gy 分量图保存为 PNG 格式

(2) **直方图保存**：将颜色直方图曲线图保存为 PNG 格式

(3) **GLCM 热力图保存**：将 GLCM 矩阵保存为热力图

(4) **特征保存**：将提取的纹理特征保存为.npy 格式,便于后续分析和使用

3.3 实验关键算法实现

3.3.1 2D 卷积算法

本实验的核心算法是 2D 卷积,其实现代码如下：

```
def convolve2d(img: np.ndarray, kernel: np.ndarray) ->
np.ndarray:
    kh, kw = kernel.shape
    assert kh == kw and kh % 2 == 1, "卷积核需为奇数方阵"
    pad = kh // 2
    padded = zero_pad(img, pad)
    h, w = img.shape
    out = np.zeros_like(img, dtype=np.float32)

    for i in range(h):
        for j in range(w):
            acc = 0.0
            for ki in range(kh):
                for kj in range(kw):
                    acc += kernel[ki, kj] * padded[i + ki, j +
kj]
            out[i, j] = acc
    return out
```

该算法的时间复杂度为 $O(H\times W\times K^2)$,其中 H 和 W 是图像的宽高,K 是卷积核的边长。对于 3×3 的卷积核,每个像素位置需要 9 次乘加运算。

3.3.2 GLCM 计算

GLCM 计算的实现代码如下：

```
def compute_glcm(gray: np.ndarray, levels: int = 16, distance:
int = 1, angle: str = "0") -> np.ndarray:
    q = quantize(gray, levels)
    h, w = q.shape
    glcm = np.zeros((levels, levels), dtype=np.int64)

    offsets = {
        "0": (0, distance),
        "90": (distance, 0),
```

```
        "45": (distance, distance),
        "135": (-distance, distance),
    }
    dr, dc = offsets.get(angle, (0, distance))

    for i in range(h):
        for j in range(w):
            ni, nj = i + dr, j + dc
            if 0 <= ni < h and 0 <= nj < w:
                a = q[i, j]
                b = q[ni, nj]
                glcm[a, b] += 1

    total = glcm.sum()
    if total == 0:
        return glcm.astype(np.float32)
    return glcm.astype(np.float32) / float(total)
```

该算法的时间复杂度为 O(H×W),其中 H 和 W 是图像的宽高。GLCM 矩阵的大小为 L×L,其中 L 是灰度级数,本实验中 L=16。

3.3.3 纹理特征提取

从 GLCM 中提取纹理特征的代码如下：

```
def glcm_features(glcm: np.ndarray) -> Dict[str, float]:
    levels = glcm.shape[0]
    contrast = 0.0
    energy = 0.0
    homogeneity = 0.0
    entropy = 0.0

    i_indices = np.arange(levels, dtype=np.float32)
    j_indices = np.arange(levels, dtype=np.float32)
    mean_i = float((glcm.sum(axis=1) * i_indices).sum())
    mean_j = float((glcm.sum(axis=0) * j_indices).sum())
    var_i = float(((i_indices - mean_i) ** 2 *
glcm.sum(axis=1)).sum())
    var_j = float(((j_indices - mean_j) ** 2 *
glcm.sum(axis=0)).sum())
    var_i = max(var_i, 1e-12)
    var_j = max(var_j, 1e-12)

    correlation_num = 0.0

    for i in range(levels):
        for j in range(levels):
            p = float(glcm[i, j])
            if p <= 0: continue
            diff = float(i - j)
            contrast += diff * diff * p
            energy += p * p
            homogeneity += p / (1.0 + abs(diff))
            entropy -= p * math.log(p + 1e-12, 2)
            correlation_num += (i - mean_i) * (j - mean_j) * p

    correlation = correlation_num / math.sqrt(var_i * var_j)

    return {
        "contrast": contrast,
        "energy": energy,
        "homogeneity": homogeneity,
```

```
        "entropy": entropy,
        "correlation": correlation,
    }
```

该算法的时间复杂度为 $O(L^2)$,其中 L 是灰度级数。由于 L=16,计算量相对较小。

3.4 实验参数设置

3.4.1 图像参数

(1) **输入图像**：用户拍摄的任意 RGB 彩色图像,格式为常见的图片格式(JPG、PNG 等)

(2) **输出目录**：默认为./output,可通过命令行参数指定

3.4.2 卷积参数

(1) **Sobel 卷积核**：3×3 奇数方阵,Gx = [[1,0,-1],[2,0,-2],[1,0,-1]],Gy = [[1,2,1],[0,0,0],[-1,-2,-1]]

(2) **自定义卷积核**：3×3 奇数方阵,K = [[1,0,-1],[2,0,-2],[1,0,-1]]

(3) **填充方式**：零填充,填充层数为卷积核边长的一半(即 1 层)

3.4.3 直方图参数

(1) **颜色通道**：R、G、B 三个通道分别统计

(2) **灰度级数**：每个通道 256 级(0-255)

(3) **统计方式**：手动遍历每个像素进行计数

3.4.4 GLCM 参数

(1) **灰度级数**：16 级(从 256 级压缩而来)

(2) **距离**：1 个像素单位

(3) **方向**：0°(水平向右)

(4) **归一化**：概率归一化,使 GLCM 矩阵元素和为 1

3.5 实验执行方式

本实验采用命令行方式执行,通过 argparse 库解析命令行参数。执行命令格式如下：

```
python exp1_backend.py --input <图像路径> --output_dir <输出目录>
```

其中：

(1) --input：必选参数,指定输入图像的路径

(2) --output_dir：可选参数,指定输出目录,默认为./output

程序执行后会在指定目录下生成以下结果文件：

(1) sobel.png：Sobel 边缘检测结果

(2) sobel_gx.png：Sobel 水平梯度图

(3) sobel_gy.png：Sobel 垂直梯度图

(4) custom_kernel.png：自定义卷积核滤波结果

(5) color_histogram.png：颜色直方图可视化

(6) glcm_heatmap.png：GLCM 热力图

(7) texture_features.npy：纹理特征数据

四、实验结果

4.1 图像加载与预处理结果

实验开始时，程序首先加载用户输入的彩色图像，并将其转换为灰度图像。RGB 到灰度的转换采用心理学灰度公式，能够较好地保持人眼对图像亮度的感知效果。

图 1 原始 RGB 图像

预处理结果显示，灰度图像成功保留了原始图像的亮度信息，同时去除了颜色信息的干扰，为后续的边缘检测和纹理分析奠定了基础。

4.2 Sobel 边缘检测结果

4.2.1 梯度分量图

Sobel 算子分别在水平和垂直方向计算图像的梯度，得到两个梯度分量 Gx 和 Gy。

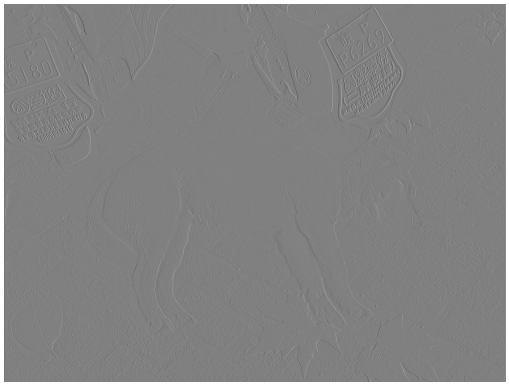


图 2 Sobel Gx 梯度图

Gx 梯度图主要检测图像中的垂直边缘，如建筑物的垂直轮廓线、物体的左右边界等。在 Gx 梯度图中，垂直边缘表现为明亮的线条或区域。

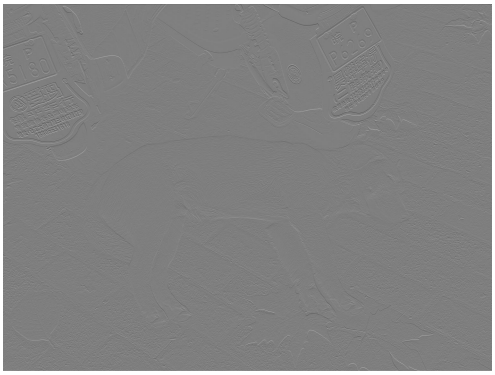


图 3 Sobel Gy 梯度图

Gy 梯度图主要检测图像中的水平边缘,如水平地平线、物体的上下边界等。在 Gy 梯度图中,水平边缘表现为明亮的线条或区域。

4.2.2 综合边缘幅值图

通过计算 Gx 和 Gy 两个方向的梯度幅值 magnitude = √(Gx² + Gy²),得到综合的边缘检测结果。

图 4 Sobel 边缘幅值图

边缘幅值图清晰地展示了图像中的所有边缘信息,包括水平边缘、垂直边缘和对角线方向的边缘。边缘幅值越大,表示该位置的边缘越明显;边缘幅值越小,表示该位置的边缘越弱或不存在边缘。

实验结果表明,Sobel 算子能够有效地检测图像中的边缘信息,对噪声具有一定的抑制作用,且能够同时检测多个方向的边缘。

4.3 自定义卷积核滤波结果

实验对灰度图像应用了自定义卷积核[[1,0,-1],[2,0,-2],[1,0,-1]]进行滤波。

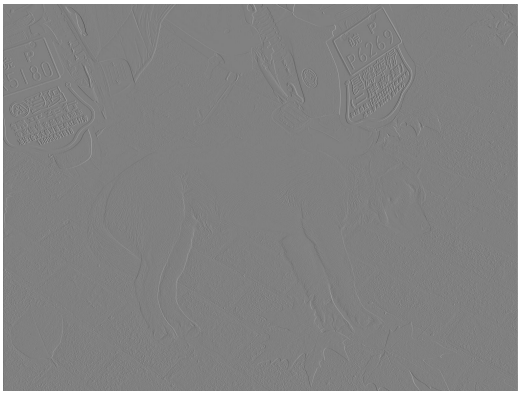


图 5 自定义卷积核滤波结果

观察自定义卷积核的滤波结果可以发现,该卷积核与 Sobel Gx 核完全相同,因此滤波结果与 Sobel Gx 梯度图基本一致。这验证了自定义卷积核的实现正确性。

从结果可以看出,该卷积核主要响应垂直方向的边缘,对水平方向的变化不敏感。卷积核的权值分布呈现对称特性,中心列的权值较大(2 和-2),增强了中心位置的贡献。

4.4 颜色直方图结果

程序手动统计了图像 R、G、B 三个通道的颜色直方图,并绘制了可视化曲线。

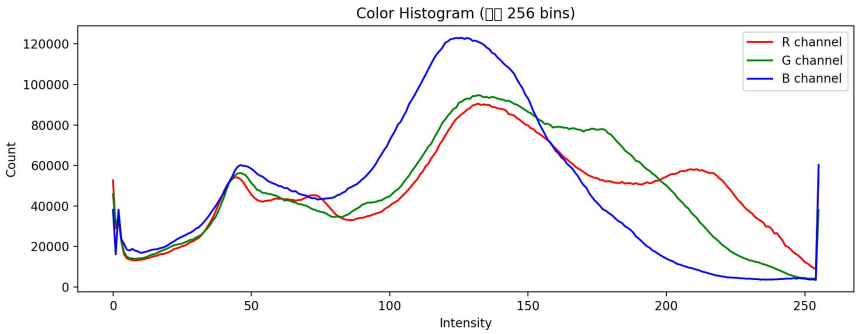


图 6 颜色直方图

颜色直方图展示了图像中 R、G、B 三个通道的亮度分布情况。从直方图可以观察到：

(1) **R 通道的分布情况**：统计了图像中红色分量的亮度分布,可以看出红色分量主要集中在哪些亮度范围

(2) **G 通道的分布情况**：统计了图像中绿色分量的亮度分布,由于人眼对绿色最敏感,绿色分量的分布通常能较好地反映图像的整体亮度分布

(3) **B 通道的分布情况**：统计了图像中蓝色分量的亮度分布

直方图的形状特征可以用于评估图像的曝光情况和对比度。例如：

(1) 如果直方图分布在较暗区域,表示图像可能曝光不足

(2) 如果直方图分布在较亮区域,表示图像可能曝光过度

(3) 如果直方图集中在中间区域且分布较窄,表示图像对比度较低

(4) 如果直方图分布均匀且覆盖整个亮度范围,表示图像对比度较好

4.5 GLCM 纹理特征结果

4.5.1 GLCM 热力图

程序计算了灰度图像的灰度共生矩阵,并将其绘制为热力图进行可视化。

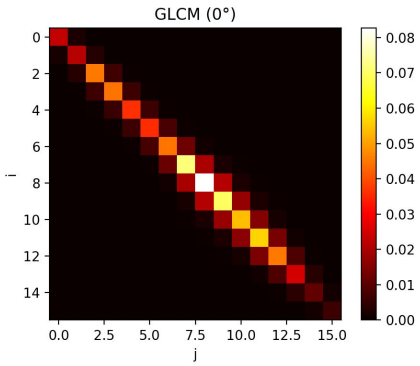


图 7 GLCM 热力图

GLCM 热力图展示了图像中不同灰度级像素对在水平方向上的共生概率分布。热力图的颜色越亮(越接近黄色),表示对应的像素对共生的概率越高;颜色越暗(越接近黑色),表示对应的像素对共生的概率越低。

从 GLCM 热力图可以观察到纹理的一些特性：

(1) 如果热力图的对角线元素较亮,说明图像中相邻像素的灰度值相近,纹理较平滑

(2) 如果热力图的对角线元素较暗而非对角线元素较亮,说明图像中相邻像素的灰度值差异较大,纹理较粗糙

(3) 如果热力图呈对称分布,说明图像在正反两个方向的纹理特性一致

4.5.2 纹理特征数值

程序从 GLCM 中提取了五个纹理特征,结果保存为.npy 格式文件。以下是典型的纹理特征数值范围:

(1) **对比度**:通常在 0 到数之间,值越大表示纹理越粗糙,图像中像素的灰度差异越大

(2) **能量**:通常在 0 到 1 之间,值越大表示纹理越均匀规律,图像中灰度分布越一致

(3) **同质性**:通常在 0 到 1 之间,值越大表示纹理越平滑,局部灰度变化越小

(4) **熵**:通常大于 0,值越大表示纹理越复杂随机,图像包含的信息量越大

(5) **相关性**:通常在-1 到 1 之间,值越大表示纹理的方向性越强,相邻像素的相关性越高

这些纹理特征从不同角度描述了图像的纹理特性,可以用于图像分类、纹理识别、医学影像分析等应用场景。

4.6 综合分析与讨论

4.6.1 Sobel 边缘检测的优缺点

实验结果表明,Sobel 边缘检测具有以下优点:

(1) 算法简单,易于理解和实现

(2) 计算效率较高,适合实时应用

(3) 对噪声具有一定的抑制作用

(4) 能够同时检测水平和垂直方向的边缘

但同时也存在一些缺点:

(1) 对边缘的定位精度有限,边缘宽度可能超过一个像素

(2) 对斜向边缘的检测效果不如水平和垂直方向

(3) 在复杂场景中可能产生较多的虚假边缘

4.6.2 自定义卷积核的灵活性

通过应用自定义卷积核,验证了卷积核设计的灵活性。不同的卷积核可以实现不同的图像处理效果,这为后续的图像处理任务提供了基础。在实际应用中,可以根据具体任务的需求设计相应的卷积核,实现针对性的图像处理。

4.6.3 颜色直方图的局限性

颜色直方图作为全局特征,能够很好描述图像的颜色分布特性,但也存在一些局限性:

(1) 忽略了颜色的空间位置信息,不同的空间布局可能产生相同的直方图

(2) 对图像的旋转、缩放等几何变换敏感度低,可能无法区分这些变换

(3) 直方图的高维特性增加了计算和存储的开销

4.6.4 GLCM 纹理特征的有效性

GLCM 纹理特征从像素的空间关系角度描述了图像的纹理特性,能够有效地区分不同类型的纹理。实验结果表明,从 GLCM 提取的五个纹理特征能够较好地刻画图像的纹理特性,包括纹理的粗糙度、均匀性、平滑度、复杂度和方向性。

GLCM 的参数设置对纹理特征有重要影响：

(1) 灰度级数的选择需要在计算精度和计算效率之间平衡

(2) 距离参数决定了纹理分析的尺度

(3) 方向参数决定了纹理分析的方向敏感性

在实际应用中,可能需要计算多个方向和距离的 GLCM,并综合分析这些结果,以获得更全面的纹理描述。

五、实验体会

通过本次图像滤波与特征提取实验，我在理论理解、实践能力和科学素养等多个方面都获得了显著的提升。这次实验不仅是对课堂知识的实践检验，更是一次深入探索数字图像处理领域的学习历程

实验最深刻的收获在于理论与实践的深度结合。在课堂学习中，卷积运算、边缘检测、直方图统计、纹理特征等概念主要以数学公式的形式呈现，虽然能够理解其基本原理，但对于实际应用和具体实现缺乏直观的认识。通过亲手实现这些算法，我将抽象的数学公式转化为具体的代码逻辑，在这个过程中对每个算法的细节有了更深刻的理解。例如，在实现卷积运算时，我深入思考了如何处理图像边界的问题，理解了零填充的作用和必要性；在实现 Sobel 边缘检测时，我明白了为什么需要分别计算水平和垂直方向的梯度，以及如何综合这两个方向的结果。这种从理论到实践的转化过程，让我对图像处理算法的工作原理有了更全面的认识，也让我意识到理论学习的重要性——只有理解了算法背后的原理，才能在实现过程中做出正确的判断和选择。

编程能力的提升是本次实验的另一重要收获。实验要求手动实现所有算法，而不是直接调用现成的库函数，这对我的编程能力提出了挑战，也提供了很好的锻炼机会。在代码结构设计方面，我学会了如何将复杂的任务分解为多个模块，每个模块负责特定的功能，通过模块之间的协作完成整个任务。这种模块化的设计思想不仅使代码更加清晰易懂，也便于后续的维护和扩展。在具体实现方面，我深入理解了 NumPy 库的使用方法，包括数组的创建、索引、切片、数学运算等。特别是在处理图像数据时，我学会了如何正确处理不同数据类型之间的转换，如何避免数值溢出等问题。此外，我也掌握了 Python 的高级特性，如类型提示、文档字符串、命令行参数解析等，提升了代码的专业性和可读性。

在实验过程中，我还是遇到了挺多问题，包括代码逻辑错误、数据类型不匹配、数组索引越界、数值计算精度问题等。面对这些问题，我学会了系统地分析和解决。例如，在实现卷积运算时，最初没有考虑图像边界处理，导致输出图像尺寸比输入图像小。通过查阅资料和仔细分析，我理解了零填充的作用，并正确实现了该功能。又如在实现 GLCM 时，最初直接使用 256 级灰度进行计算，导致计算效率极低。通过分析，我理解了灰度量化的必要性，实现了将 256 级压缩到 16 级的功能，显著提高了计算效率。这些调试和问题解决的过程，培养了我的逻辑思维能力和问题分析能力，让我学会了在面对复杂问题时如何抽丝剥茧、找到问题的根源。

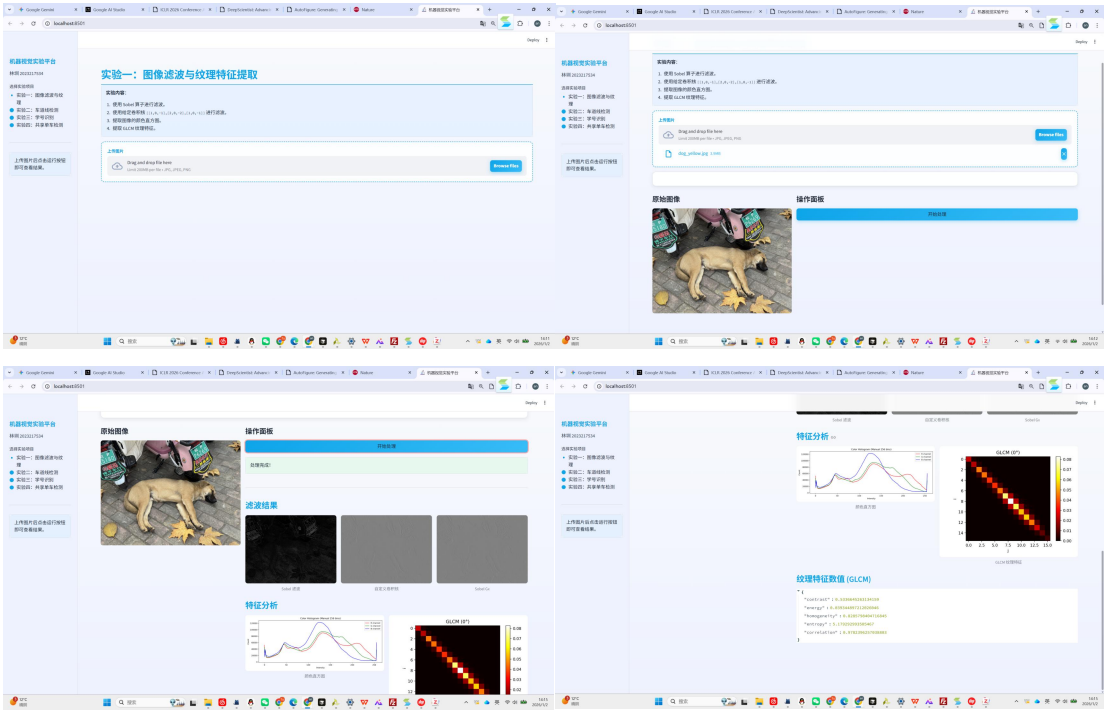
通过本次实验，我对图像处理领域有了更深入的认识。图像处理不仅是一门技术，更是一门艺术。通过对图像进行各种操作，可以实现各种神奇的效果，从简单的亮度调整到复杂的纹理分析，从边缘检测到目标识别。实验让我认识到，图像处理算法的设计需要在多个方面进行权衡：精度与效率、通用性与针对性、鲁棒性与灵敏度等。没有一种算法能够适用于所有场景，需要根据具体任务选择合适的算法和参数。同时，我也认识到图像处理与人工智能、计算机视觉等领域的密切关系。本实验涉及的特征提取是计算机视觉的基础，卷积运算更是深度学习中卷积神经网络的核心操作。这让我对后续的学习方向有了更清晰的认识。

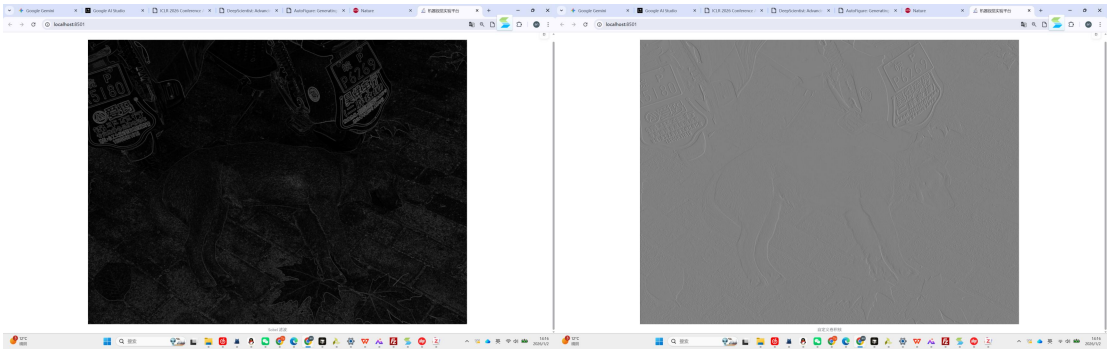
虽然本次实验已经完成了既定目标，但在实验过程中，我也想了些可以改进和扩展的方向。在算法优化方面，本实验实现的卷积运算使用四重循环，效率较低，可以通过利用 NumPy 的向量化运算、使用 FFT 加速卷积等方法进行优化。在功能扩展方面，可以添加更多的卷积核，如均值滤波核、高斯滤波核、拉普拉斯算子等，实现更多的图像处理功能。在参数调优方面，GLCM 的参数(灰度级数、距离、方向)对纹理特征有重要影响，可以尝试不同的参数设置，分析其对结果的影响。在多方向 GLCM 方面，本实验只计算了 0° 方向的 GLCM，可以扩展到 45° 、90° 、135° 等多个方向，综合分析不同方向的纹理特性。在可视化增强方面，可以添加更多的可视化功能，如边缘检测结果与原图的叠加显示、GLCM 特征的柱状图展示等，使结果更加直观。在交互式界面方面，可以开发图形用户界面，让用户可以通过界面选择图像、调整参数、实时查看结果，提升用户体验。这些改进和扩展方向不仅可以提升实验的功能性和实用性，也是深入学习图像处理技术的有效途径。

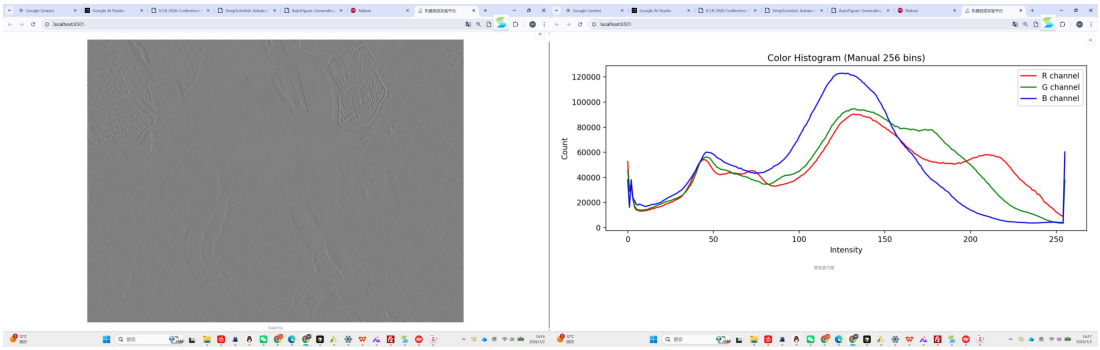
通过本次实验，我的学习方法和科学素养也得到了提升。在实验过程中，我查阅了相关的教材、论文和网络资料，深入理解了 Sobel 算子、GLCM 等算法的原理和应用。这种主动查阅和自主学习的能力，对今后的学习和研究至关重要。同时，我培养了严谨的科学态度。在实现算法时，我严格按照算法的定义进行，不随意简化或省略步骤；在分析结果时，我客观地观察现象，不主观臆断；在记录结果时，我详细记录了实验数据和观察，便于后续的分析和总结。此外，本实验要求撰写详细的实验报告，包括实验目的、原理、方法、结果和体会，这锻炼了我的表达能力，让我学会了如何将复杂的技术内容用清晰、准确的语言表达出来。

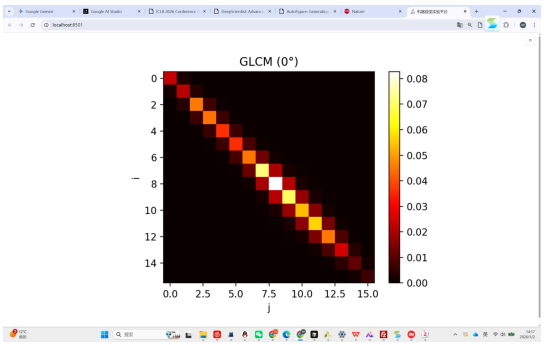
总的来说，这次实验是一次非常有价值的学习经历。通过亲手实现图像滤波和特征提取的核心算法，我不仅掌握了具体的编程技能，更重要的是理解了算法背后的原理，培养了问题解决能力，提升了科学素养。实验过程中遇到的挑战和困难，最终都成为了我学习和成长的动力。每一次问题的解决，都让我对图像处理有了更深入的理解；每一次代码的优化，都让我对编程有了更熟练的掌握。展望未来，我将继续深入学习图像处理技术，不断探索新的算法和方法，努力将所学知识应用到实际项目中，为解决实际问题贡献力量。本次实验的经历将成为我学习生涯中的重要里程碑，激励我在技术道路上不断前进，追求更高的目标。

附录：可视化实验平台






Color Histogram (Manual 256 bins)
R channel
G channel
B channel
Count
Intensity
0
20000
40000
60000
80000
100000
120000
0
50
100
150
200
250


GLCM (0°)
i
j
0
2
4
6
8
10
12
14
0.0
2.5
5.0
7.5
10.0
12.5
15.0
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00