



**机器视觉实验报告**

|  |  |
| --- | --- |
| 专 业 班 级 | 智能科学与技术22-1班 |
| 学生姓名及学号 | 黄聖翔 2022212383 |
| 课程教学班号 | 0571122B-001 |
| 任 课 教 师 | 洪日昌、吴晶晶 |
| 实验地点 | 科教楼C座201 |

2024-2025学年第一学期

# 实验一：图像滤波

## 实验目的

本实验旨在通过手动实现图像处理中的滤波操作、颜色直方图提取以及纹理特征提取，深入理解图像处理的基本原理和方法。具体目标包括：

* **掌握Sobel算子的原理与应用**：通过手动实现Sobel算子，理解边缘检测的基本原理。
* **理解卷积操作的作用**：通过给定的卷积核进行滤波操作，掌握卷积在图像处理中的应用。
* **学习颜色直方图的提取**：通过计算图像的颜色直方图，理解图像的颜色分布特征。
* **掌握纹理特征的提取方法**：通过手动实现纹理特征提取，理解图像纹理的基本概念及其在图像分析中的应用。
* **提高编程能力**：通过手动实现图像处理算法，提升编程能力，尤其是对图像数据的处理能力。

## 实验原理

### **Sobel算子**

Sobel算子是一种经典的边缘检测算子，通过计算图像像素点邻域的梯度来检测边缘。它使用两个3×3的卷积核分别计算水平方向（x方向）和垂直方向（y方向）的梯度。

水平方向卷积核为，用于检测垂直边缘；

垂直方向卷积核为，用于检测水平边缘。

通过计算这两个方向梯度的平方和的平方根，可以得到最终的边缘强度图像。Sobel算子对噪声具有平滑作用，能够提供较为准确的边缘方向信息。

### **卷积操作**

卷积操作是图像处理中的基础运算，其本质是用一个卷积核对输入图像进行加权求和。卷积核是一个通常为奇数大小的矩阵，在进行卷积时，将卷积核的中心对准输入图像的每个像素，将卷积核覆盖区域内的像素值与卷积核对应位置的值相乘并求和，得到的结果作为输出图像对应位置的像素值。在本实验中，除了Sobel算子外，还使用了给定的卷积核进行图像滤波。卷积操作可以实现图像的模糊、锐化、边缘检测等多种效果，是图像处理中最重要的操作之一。

### **非极大值抑制和滞后边缘跟踪**

非极大值抑制和滞后边缘跟踪应用于Sobel边缘检测中用于优化边缘检测结果。在Sobel算子计算出梯度幅值和方向后，这两个步骤可以帮助细化边缘并提高边缘的连续性。非极大值抑制通过比较每个像素在梯度方向上的邻域像素值，保留局部最大值点。具体来说，将梯度方向量化为四个主要方向（0°、45°、90°、135°），然后将当前像素与其梯度方向上的两个相邻像素进行比较。如果当前像素的梯度幅值不是局部最大值，则将其抑制为0。

滞后边缘跟踪则通过设置高低两个阈值（本实验中高阈值为图像最大梯度值的15%，低阈值为高阈值的5%），对非极大值抑制后的结果进行进一步处理。高于高阈值的像素被直接确定为边缘点，低于低阈值的像素被直接抑制，而介于两个阈值之间的像素只有在与确定的边缘点相连时才被保留为边缘点。这种双阈值的处理方法可以有效地抑制噪声，同时保持边缘的连续性。

这两个步骤的引入显著提升了Sobel边缘检测的效果。非极大值抑制帮助细化边缘至一个像素宽，避免了边缘的"粗化"现象；滞后边缘跟踪则通过智能地连接边缘点，提高了边缘的连续性，同时有效地抑制了噪声。这种改进的Sobel边缘检测方法不仅保留了Sobel算子对噪声不敏感的优点，还获得了更加清晰和准确的边缘检测结果。

### **颜色直方图**

颜色直方图是描述图像中像素值分布情况的统计图表，它将图像中的像素值进行提取并将二维分布转化为一维分布展示在直方图中，统计了图像中每个灰度级或颜色分量出现的频率。对于RGB彩色图像，需要分别计算红、绿、蓝三个通道的直方图。直方图的横轴表示像素值（0-255），纵轴表示具有该像素值的像素个数。通过分析颜色直方图，可以了解图像的明暗分布、对比度等特征，是图像处理和分析中的重要工具。颜色直方图具有旋转不变性，但对光照变化较为敏感。

### **纹理特征提取**

纹理特征提取采用灰度共生矩阵（GLCM）方法，该方法通过分析图像中像素对的空间关系来描述图像的纹理特征。GLCM统计了在特定距离和方向上，两个像素点的灰度值共同出现的频率。

基于GLCM可以计算多种纹理特征，本实验计算了对比度（反映图像的清晰度和纹理的深浅程度）、同质性（反映图像局部变化的程度）和能量（反映图像灰度分布的均匀性和纹理的粗细）三个特征。这些特征能够有效地描述图像的纹理信息，广泛应用于图像分类、识别等任务中。GLCM的优势在于它不仅考虑了像素值的统计分布，还考虑了像素值的空间相关性，因此能够更好地描述图像的纹理特征。在实际应用中，GLCM特征常与其他特征（如颜色特征、形状特征等）结合使用，以提供更全面的图像描述。此外，通过选择不同的距离和方向参数，可以捕捉不同尺度和方向上的纹理信息，使特征提取更加灵活和有效。

## 实验方法

主函数process\_image实现了完整的图像处理流程如图3-1所示。程序首先读取指定路径的输入图像，并将其转换为灰度图像用于后续处理。随后，程序并行执行四个主要的处理任务。在Sobel边缘检测部分，首先计算水平和垂直方向的Sobel梯度，然后计算梯度幅值和方向，接着通过非极大值抑制细化边缘，最后使用双阈值检测和滞后边缘跟踪得到最终的边缘检测结果。自定义卷积核滤波通过应用给定的卷积核进行图像滤波，并对结果进行二值化处理。颜色特征分析部分分别计算图像的RGB三通道直方图和灰度直方图，反映图像的颜色分布特征。纹理特征提取则通过计算灰度共生矩阵（GLCM），并从中提取对比度、同质性和能量三个特征值，将结果保存为npy格式文件。最后，程序将所有处理结果进行显示和保存，完成整个图像处理流程。

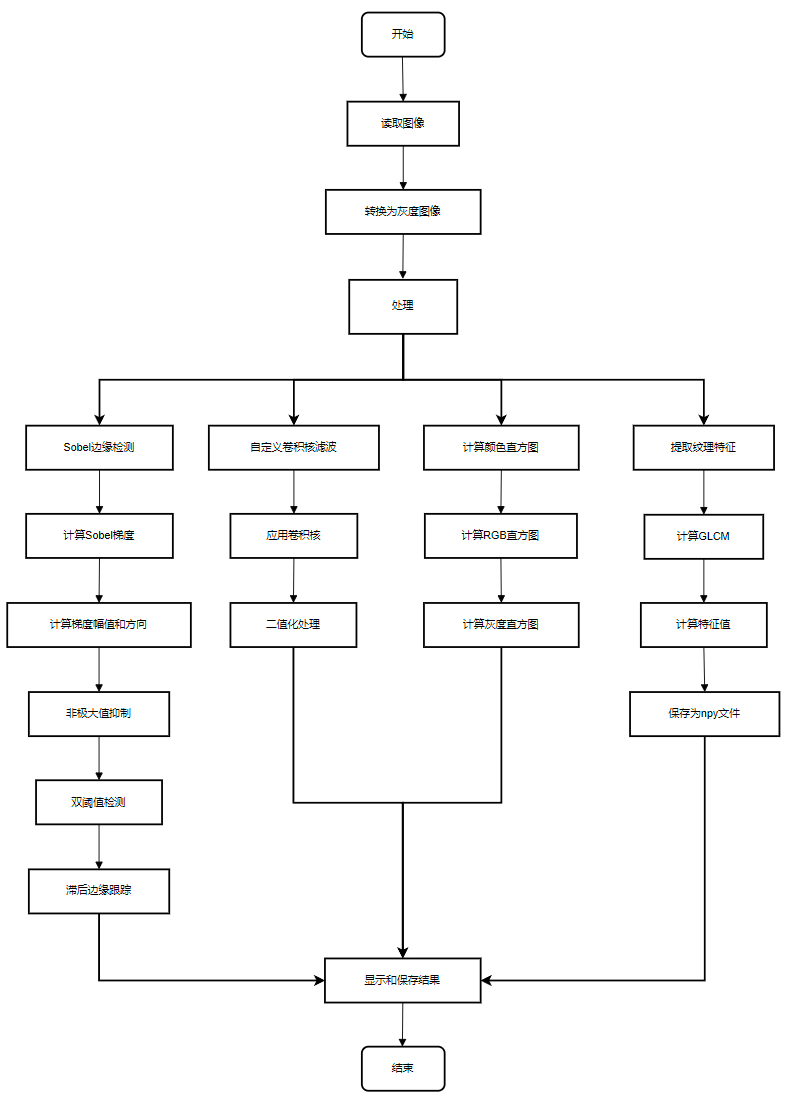


图3-1 程序主要流程图

### **Sobel滤波**

Sobel滤波的实现定义了两个3×3的Sobel算子，分别用于检测水平和垂直方向的边缘。水平方向的Sobel算子用于检测垂直边缘，垂直方向的Sobel算子用于检测水平边缘。通过自定义的卷积函数分别计算图像在这两个方向上的梯度，然后计算梯度幅值和方向。随后使用非极大值抑制技术对梯度幅值图像进行处理，通过比较像素在梯度方向上的邻域值，保留局部最大值点，从而将边缘细化到一个像素宽。随后，采用双阈值检测方法，设置高低两个阈值（高阈值为图像最大梯度值的15%，低阈值为高阈值的5%），初步确定边缘像素。最后，通过滞后边缘跟踪技术，将与强边缘点相连的弱边缘点也确定为边缘，从而得到连续的边缘检测结果。这种改进的Sobel边缘检测方法不仅保持了边缘的清晰度，还提高了边缘的连续性。

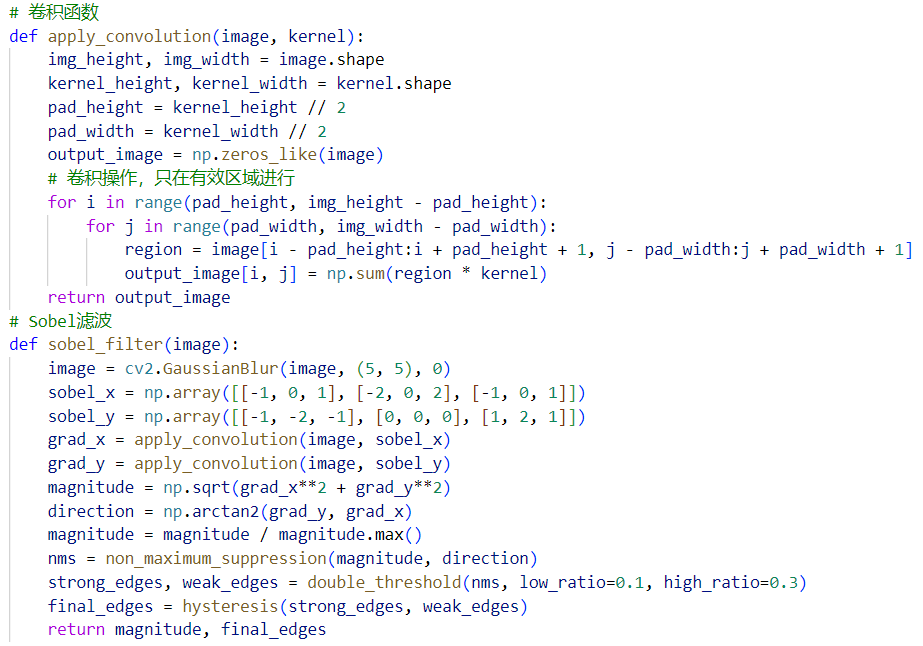


图3-2 Sobel滤波与非极大值抑制代码

### **给定卷积核滤波**

使用题目指定的卷积核对图像进行滤波处理。这个卷积核的结构与Sobel算子的水平方向核相似，同样具有检测垂直边缘的作用。通过相同的卷积函数对图像进行处理，得到滤波结果。与Sobel滤波相比，这个卷积核在边缘检测时对垂直边缘的响应更强，而对水平边缘的响应较弱，因此能够突出显示图像中的垂直边缘特征。为了使边缘特征更加明显，对滤波结果进行二值化处理，通过设置合适的阈值将图像转换为黑白二值图像。这种处理方法能够清晰地展示图像中的垂直边缘结构，为图像的边缘分析提供了不同的视角。通过比较两种滤波方法的结果，可以观察到它们在边缘检测效果上的差异，特别是在处理不同方向边缘时的表现。

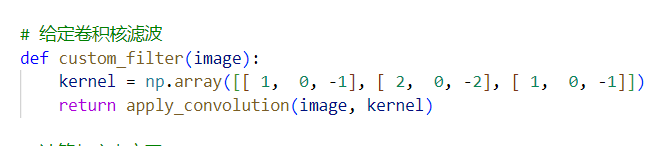


图3-3 给定卷积核滤波

### **提取颜色直方图**

颜色直方图的提取分为两个部分：灰度直方图和RGB颜色直方图。对于灰度直方图，通过遍历图像的每个像素，使用一个长度为256的数组记录每个灰度级别的像素数量。对于RGB颜色直方图，需要分别计算图像红、绿、蓝三个通道的直方图。程序创建了三个独立的直方图数组，分别统计每个通道中256个灰度级别的像素数量。使用Matplotlib库将这四个直方图（一个灰度直方图和三个RGB通道直方图）绘制在同一个图形中，采用1×4的布局进行显示，使得直方图的对比更加直观。通过分析这些直方图，可以了解图像的明暗分布、色彩特征和对比度等信息。



图3-4 直方图计算函数

### **提取纹理特征并保存**

纹理特征提取采用灰度共生矩阵（GLCM）方法，通过分析图像中相邻像素对的灰度值关系来描述图像的纹理特征。首先创建一个256×256的零矩阵作为GLCM，然后遍历图像，统计相邻像素对（在本实验中选择右下方向的相邻像素）的灰度值出现频率。基于构建的GLCM矩阵，计算三个重要的纹理特征：对比度、同质性和能量。对比度通过计算像素对灰度差值的平方与出现频率的乘积得到，反映图像的清晰度；同质性通过计算像素对灰度差值的倒数与出现频率的乘积得到，反映图像的局部均匀性；能量则是GLCM矩阵元素平方和，反映图像纹理的规则性。最后，将这三个特征值保存为npy格式文件，便于后续分析使用。这种基于GLCM的纹理特征提取方法能够有效地描述图像的纹理信息，为图像分析提供重要的特征数据。

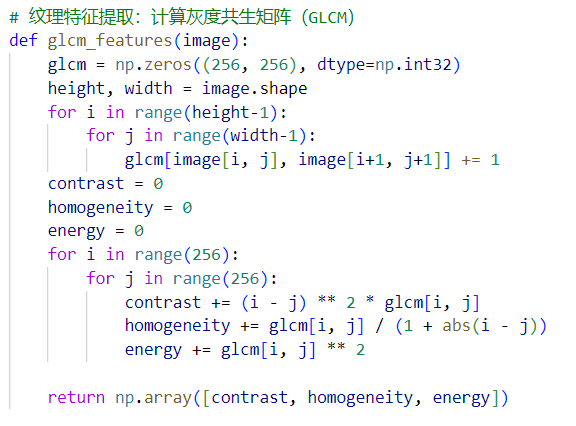


图3-5 提取纹理特征函数

## 实验结果

### **滤波结果**

本实验对自拍摄的图像分别进行了Sobel边缘检测和给定卷积核滤波处理。在Sobel边缘检测中，通过结合非极大值抑制和滞后边缘跟踪技术，得到了清晰且连续的边缘检测结果。从图4-1(b)(c)可以观察到，改进后的Sobel算法能够有效地检测出图像中的主要边缘结构，包括物体的轮廓和内部的显著边缘特征。边缘线条细化到一个像素宽，且具有良好的连续性，这得益于非极大值抑制的细化作用和滞后边缘跟踪的连接效果。使用给定卷积核进行滤波的结果如图4-1(d)所示。与Sobel滤波相比，该卷积核对垂直边缘的响应更为突出，这可以从检测结果中清晰地观察到。垂直方向的边缘特征被很好地保留和增强，而水平方向的边缘则相对较弱。这种特性使得该滤波方法特别适合于检测和分析图像中的垂直结构。

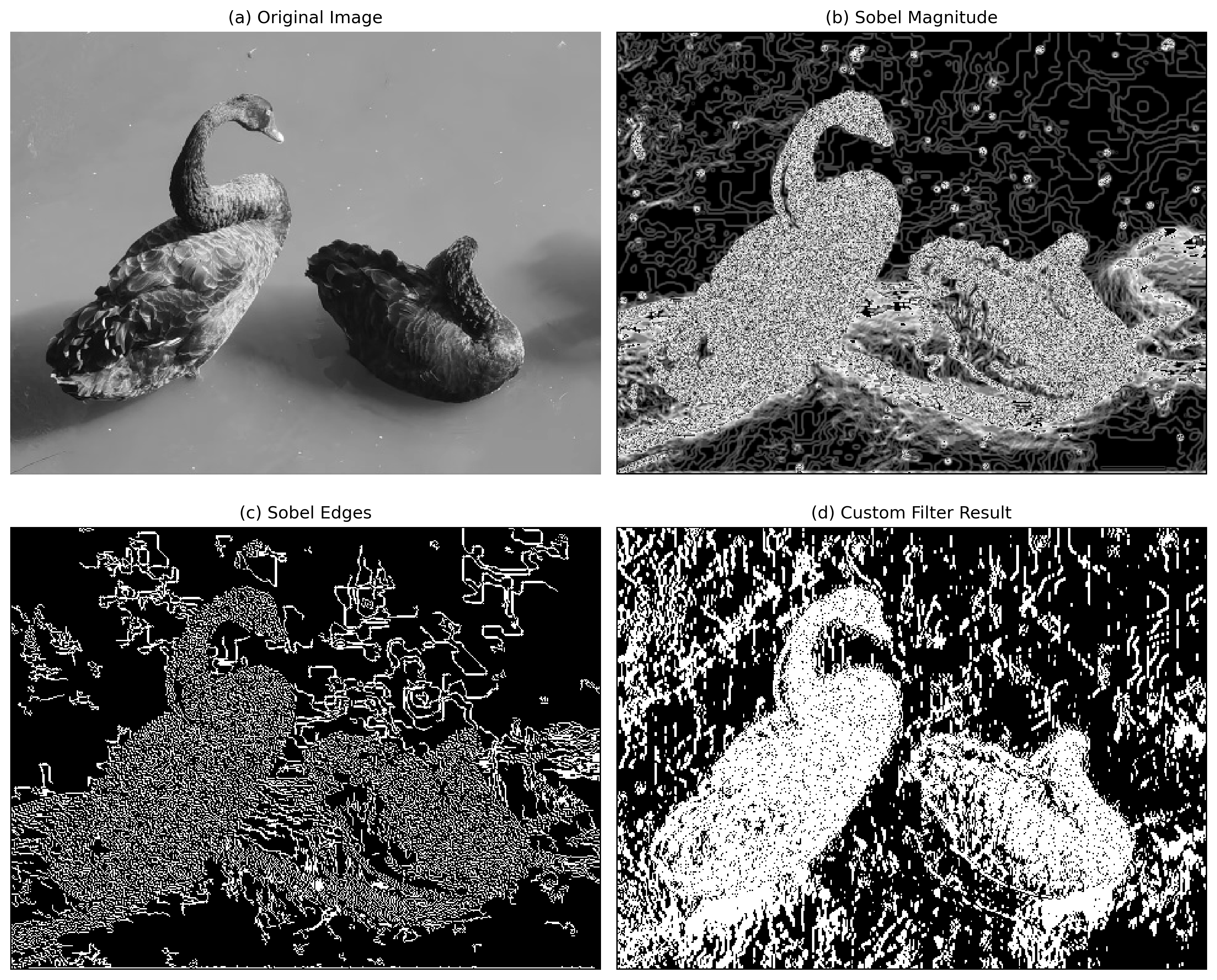


图4-1 滤波结果

然而，在实验过程中也观察到一些滤波效果不够理想的情况，主要原因有两个方面：原图存在较大的噪声干扰。如图4-1(a)所示，我们希望能够通过Sobel滤波提取出尽可能好的边缘图像，但是图像中不同的细节变化导致了图像在多个地方具有较高的响应，所以经过滤波器以及非极大值抑制等后仍有一定的噪声；其次，当前的参数设置还需要进一步优化。非极大值抑制中的梯度方向量化可能过于粗糙，双阈值检测中的高低阈值比例（当前为0.3和0.1）也可能需要根据具体图像特征进行调整。这些参数的设置直接影响了边缘检测的质量和准确性。

### **直方图结果**

颜色直方图分析结果如图4-2所示，包括RGB三个通道的直方图和灰度直方图。从RGB直方图可以观察到，红色通道的像素值分布较为集中像素值120-150范围；绿色通道的像素值分布较为集中像素值150-170范围；蓝色通道像素值分布在50-100范围。这种分布特征反映了。灰度直方图展示了图像的整体明暗分布特征。从图中可以看出，灰度值主要集中在中间区域（像素值150-180范围），且呈现出多个局部峰值，这表明图像具有丰富的灰度层次，对比度适中。

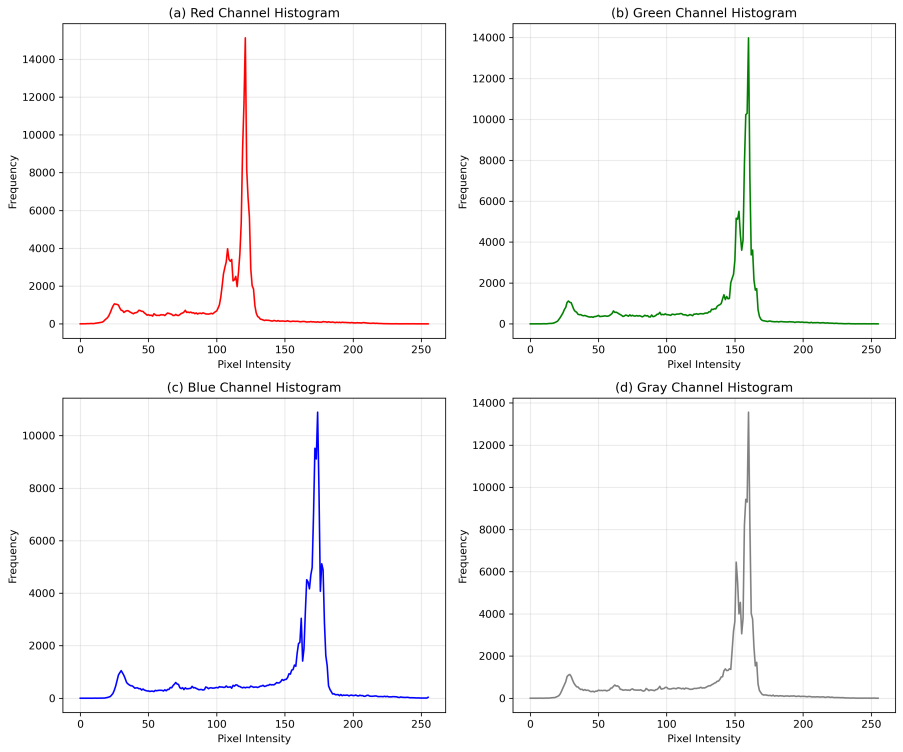


图4-2 提取图像颜色直方图

### **灰度共生矩阵结果**

通过计算灰度共生矩阵保存了图像的纹理特征并将其可视化为如图4-3所示的热力图结果。该热力图采用对数尺度显示，以更好地展示矩阵中的值分布。从图中可以观察到一条明显的对角线结构，呈现为从亮红色到黄色的渐变，这表明图像中存在大量灰度值相似的相邻像素对。对角线的强度在灰度值50-150范围内最为显著，这与图像的主要灰度分布区域相对应，说明图像的灰度值主要集中在中间范围。对角线的宽度较窄且亮度分布集中，表明图像的纹理变化相对平滑。在对角线周围可以观察到较弱的红色分布，这反映了图像中存在渐进的灰度变化，而不是突变的纹理特征。离对角线较远的区域呈现出深色，说明图像中很少出现灰度值差异较大的相邻像素对，这进一步证实了图像纹理的平滑特性。矩阵整体呈现出良好的对称性，这是GLCM的典型特征，同时背景区域（深色区域）占据了矩阵的大部分，表明大多数灰度值组合在图像中出现的频率很低。

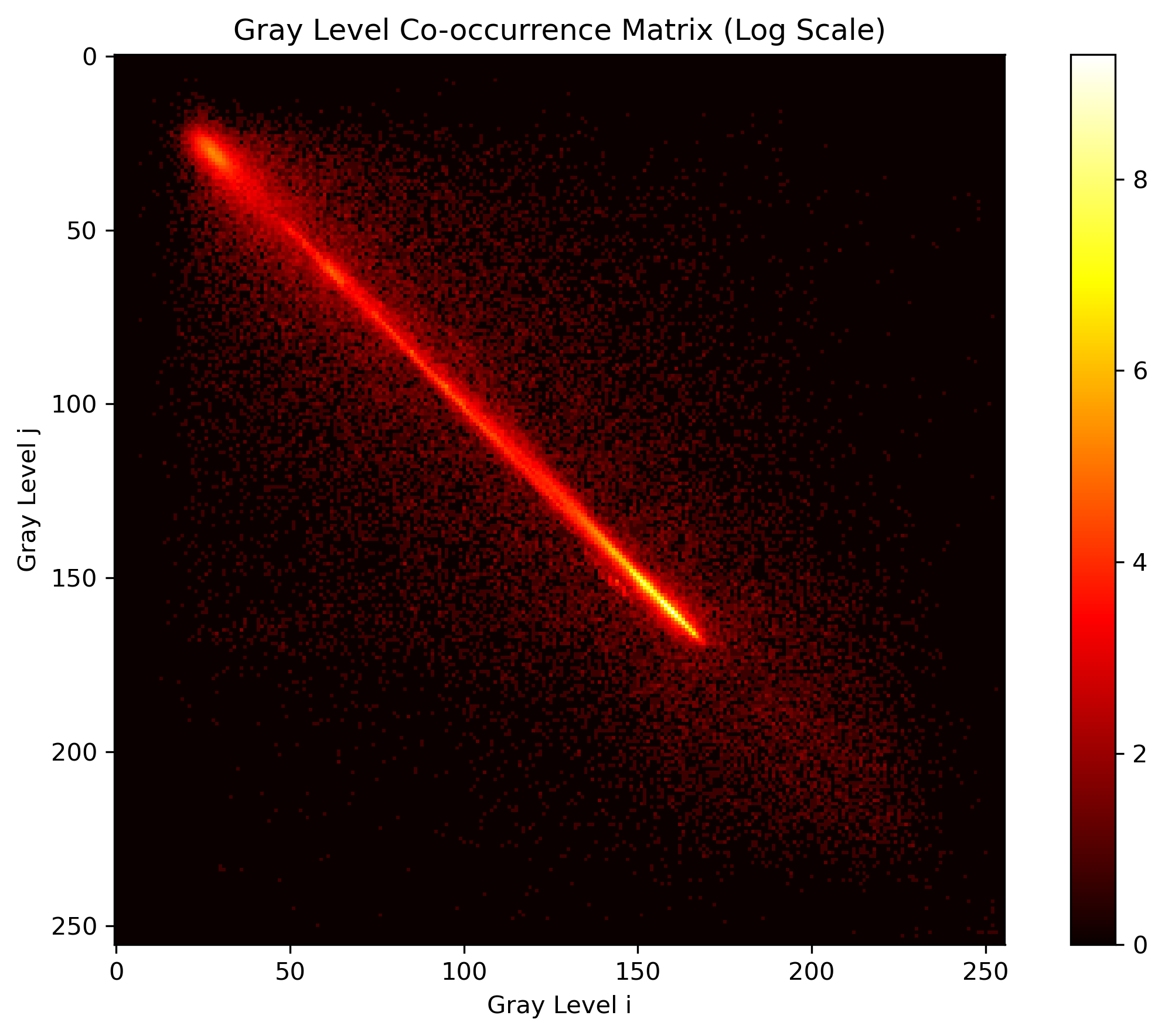


图4-3 纹理特征热力图可视化

## 实验体会

通过本次图像处理实验，我深入理解了图像边缘检测、直方图分析和纹理特征提取的基本原理和实现方法。在实验过程中，我不仅掌握了Sobel算子和自定义卷积核进行边缘检测的技术，还学会了如何通过非极大值抑制和滞后边缘跟踪来优化边缘检测结果。这些技术的实现让我对图像处理的基础算法有了更深刻的认识，特别是理解了卷积操作在图像处理中的重要作用。

在编程实现过程中，我遇到了一些挑战，比如如何处理图像边界问题、如何选择合适的阈值参数等。这些问题的解决过程让我学会了如何将理论知识转化为实际的程序代码，同时也培养了调试和优化程序的能力。特别是在处理图像噪声问题时，我认识到了预处理步骤的重要性，以及参数选择对最终结果的影响。

通过分析RGB颜色直方图和灰度直方图，我学会了如何从统计的角度来理解图像的颜色和亮度分布特征。直方图分析不仅帮助我理解了图像的整体特征，还为图像增强和阈值选择提供了重要依据。而在实现灰度共生矩阵（GLCM）计算时，我对图像纹理特征的量化描述有了更深的理解。GLCM的计算和特征提取过程让我认识到，图像的纹理信息可以通过数学方法进行精确描述和分析，这对于图像识别和分类具有重要意义。

# 实验二：车道线检测

## 实验目的

* 掌握基于霍夫变换的车道线检测算法原理和实现方法
* 理解图像预处理技术在车道线检测中的重要作用
* 学习边缘检测、形态学处理等图像处理基础技术
* 掌握霍夫变换在直线检测中的应用

## 实验原理

### **2.1图像预处理**

图像预处理是车道线检测的基础步骤，对后续处理的效果有重要影响。首先需要将输入的彩色图像转换为灰度图像，这样可以减少数据维度，简化计算过程。然后使用高斯滤波器对图像进行平滑处理，高斯滤波通过对图像进行加权平均，可以有效地抑制图像中的高频噪声，同时保持图像的主要结构特征。高斯核的大小和标准差是重要的参数，需要根据图像特征进行调整。接下来使用Canny边缘检测算子提取图像中的边缘信息。Canny算子首先计算图像的梯度幅值和方向，然后通过非极大值抑制细化边缘，最后通过双阈值检测和边缘跟踪确定最终的边缘。这种多步骤的边缘检测方法可以得到连续且准确的边缘。

### **2.2形态学处理**

在本实验中，主要使用膨胀和腐蚀这两种基本的形态学操作。膨胀操作使用结构元素对图像进行扫描，当结构元素的中心像素遇到目标像素时，结构元素覆盖的区域都被设为目标像素。这样可以扩大边缘区域，连接断裂的边缘线段。腐蚀操作则是膨胀的对偶操作，当结构元素完全位于目标区域内时，中心像素被保留为目标像素。腐蚀操作可以去除细小的噪声点和非车道线的边缘特征。通过合理设计结构元素的形状和大小，并结合膨胀和腐蚀操作，可以有效地增强车道线的边缘特征，同时抑制干扰信息。

### **2.3霍夫变换原理**

在图像空间中，一条直线可以用参数方程表示，其中是直线到坐标原点的垂直距离，θ是垂线与x轴的夹角。对于图像中的每个边缘点，都可以在参数空间中绘制一条正弦曲线，表示经过该点的所有可能直线。当多个边缘点位于同一直线上时，它们在参数空间中的正弦曲线会在同一点相交。通过在参数空间中建立累加器数组，统计每个(ρ,θ)组合获得的投票数，可以找到参数空间中的局部最大值点，这些点对应图像中的主要直线。霍夫变换的优势在于它对噪声和部分遮挡具有较强的鲁棒性，能够有效地检测出图像中的直线结构。

### **直线筛选与合并**

直线筛选与合并是对霍夫变换检测结果进行后处理的重要步骤。由于霍夫变换可能检测到大量的直线，其中包括许多不属于车道线的直线，因此需要进行筛选。首先根据直线的斜率进行筛选，车道线通常具有特定的斜率范围，可以排除那些过于水平或垂直的直线。然后根据直线的长度进行筛选，过短的线段通常是噪声或无关的边缘。对于通过初步筛选的直线，需要进行合并处理。相近的直线可能来自同一条车道线的不同部分，通过计算直线之间的角度差和距离，可以将这些直线合并。合并时可以采用加权平均的方式，权重可以基于直线的长度或投票数。最后，将合并后的直线分为左右车道线，通过对每侧的直线进行平均或拟合，得到最终的车道线位置。这种多步骤的后处理可以显著提高车道线检测的准确性和可靠性。

## 实验方法

整个实验流程主要分为四个关键阶段：图像预处理与形态学处理、霍夫变换实现、直线筛选与合并，以及结果可视化。通过这些步骤的有机结合，实现了从原始图像输入到最终车道线检测结果输出的完整处理流程。每个阶段都针对特定的处理目标，采用了相应的算法和优化策略，确保检测结果的准确性和可靠性。

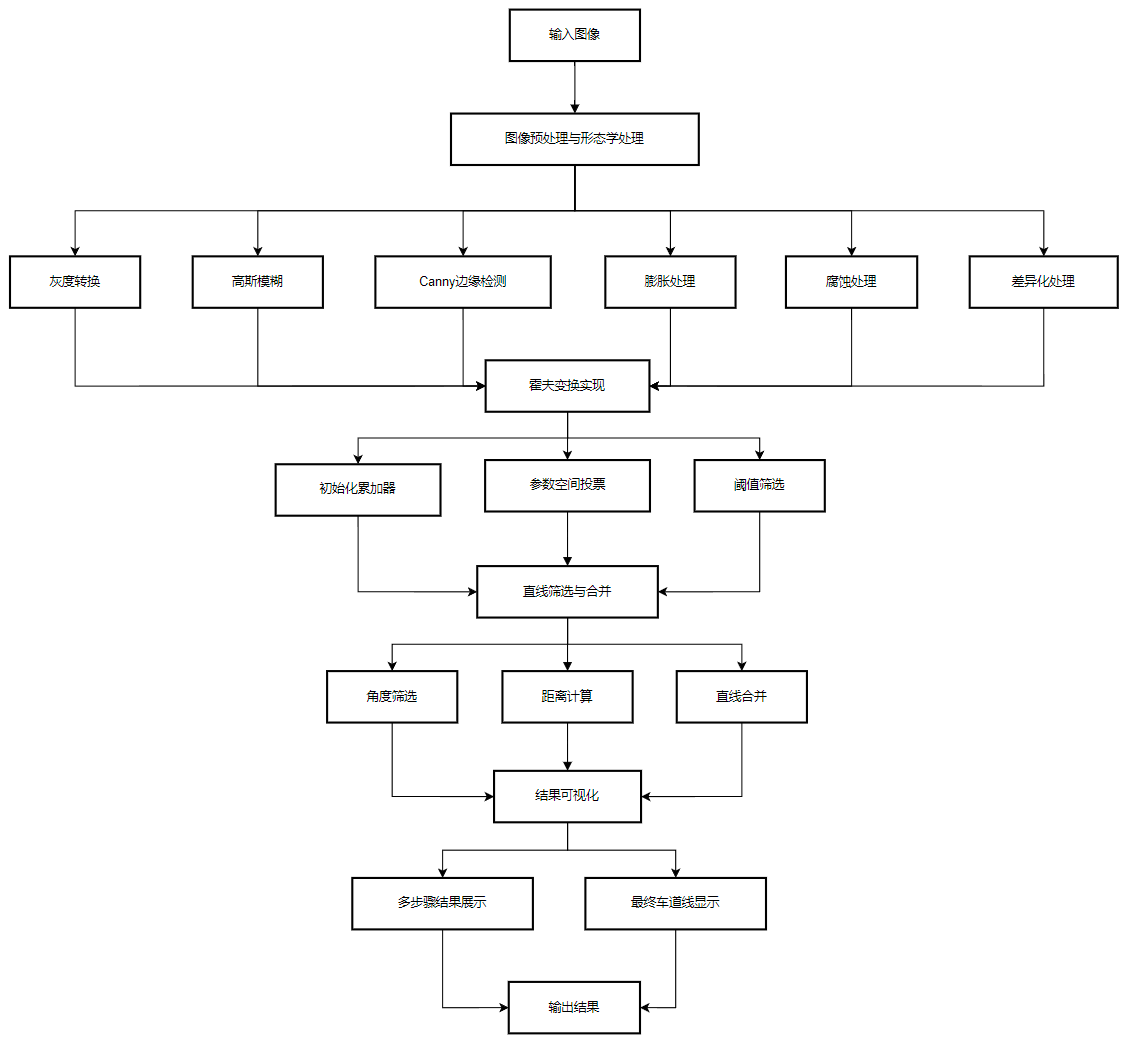


图3-1 实验流程图

### **3.1图像预处理与形态学操作**

本实验的第一阶段包括图像预处理和形态学处理两个密切相关的步骤。图像预处理阶段使用OpenCV库对输入图像进行基础处理。通过cv2.imread()函数读取输入的道路图像，考虑到车道线检测主要关注图像的结构特征而非颜色信息，使用cv2.cvtColor()函数将图像转换为灰度图像，这样可以降低数据维度并简化后续处理。为了减少图像中的噪声干扰，应用cv2.GaussianBlur()函数进行高斯模糊处理并使用cv2.Canny()函数进行边缘检测，通过设置低阈值50和高阈值150，可以有效地检测出图像中的显著边缘，同时抑制虚假边缘的产生。

在获得边缘检测结果后，立即进行形态学处理以优化边缘特征。使用numpy库创建一个3×3的矩形结构元素，这个结构元素由全1矩阵组成，作为形态学操作的基本单元。膨胀操作使用cv2.dilate()函数实现，设置迭代次数为10，这样可以有效地扩展边缘区域，连接由于噪声或光照变化导致的边缘断裂。膨胀操作的本质是取结构元素覆盖区域内的最大值，因此可以填充边缘周围的空隙。随后的腐蚀操作使用cv2.erode()函数，迭代次数设为20，用于去除噪声点和细小的非车道线边缘特征。腐蚀操作取结构元素覆盖区域的最小值，可以有效地消除孤立的噪声点。最后使用cv2.subtract()函数计算原始边缘图像与腐蚀结果的差异，这一步可以突出显示主要的边缘特征，同时抑制背景噪声。这种差异化处理的结果为后续的霍夫变换提供了更清晰的输入图像。预处理和形态学处理的组合不仅提高了边缘检测的质量，也为后续的车道线检测奠定了良好的基础。通过这些处理步骤，我们可以有效地突出图像中的车道线特征，同时抑制其他干扰信息，提高后续霍夫变换的准确性。



图3-2 图像预处理与形态学操作

### **3.2霍夫变换原理**

霍夫变换的实现是本实验中最核心的部分，采用了参数空间投票的方式来检测直线。首先获取差异化处理后图像的尺寸信息，并计算可能的最大半径值，这个值通过图像宽度和高度的平方和开方得到，确保能够覆盖图像中任意位置的直线。同时初始化一个二维累加器数组，其大小为(2 \* r\_max, 180)，用于存储参数空间中的投票结果。数组的第一维表示距离参数，考虑到可能为负值，因此大小设为最大半径值的两倍；第二维表示角度参数，范围从0到179度。

在投票过程中，遍历图像中的每个边缘点，对于每个点，在0到179度范围内计算所有可能的直线参数。通过公式计算每个角度对应的距离值，并在累加器相应位置增加计数。为了处理负的距离值，在存储时将计算得到的r值加上r\_max进行偏移。设置阈值100用于过滤投票数较少的直线，只保留得票数超过阈值的参数组合并将这些筛选出来的组合存放在列表中，这些参数对应图像中的主要直线特征。对于每个超过阈值的位置，将参数转换回笛卡尔坐标系，计算直线的端点坐标，并使用cv2.line()函数在原图上绘制检测到的直线。

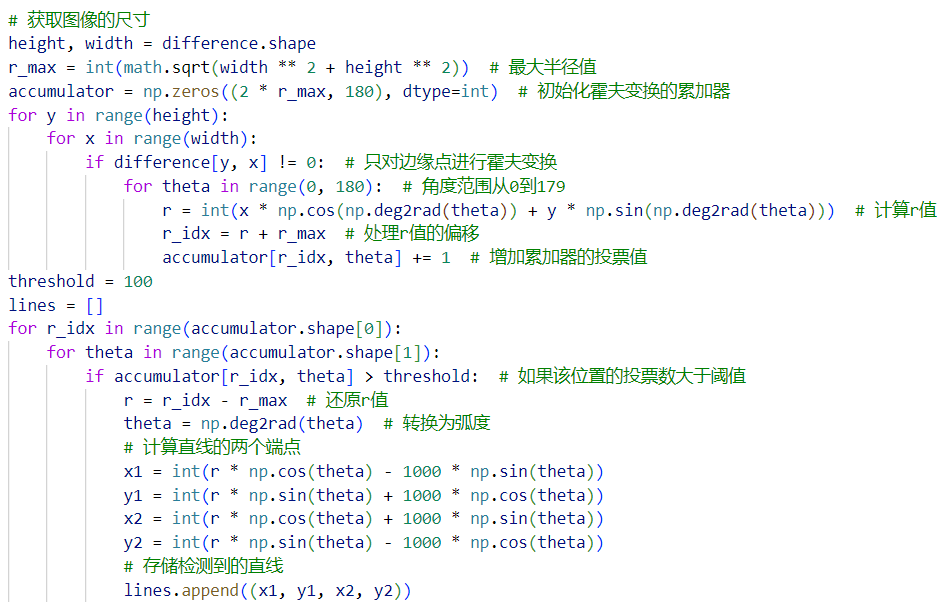


图3-3 霍夫变化投票函数

### **3.3直线筛选与合并**

直线筛选与合并阶段设计了一个专门的merge\_lines函数来优化霍夫变换的检测结果。该函数接受两个重要参数：角度阈值（angle\_threshold）设置为20度，用于判断两条直线是否具有相似的方向；距离阈值（distance\_threshold）设置为30像素，用于判断两条直线是否足够接近。函数的处理流程是遍历所有检测到的直线，对于每条直线，计算其与已保存直线的角度差和距离。角度差通过计算两条直线的斜率，然后使用反正切函数得到角度值；距离则通过点到直线的距离公式计算。当两条直线的角度差小于阈值且距离小于阈值时，认为它们属于同一条车道线，应该被合并。合并操作通过取两条直线端点的最小和最大坐标值来创建一条新的直线，这样可以得到一条更长、更准确的车道线。这种合并策略可以有效地减少重复检测的直线，提高车道线检测的准确性。

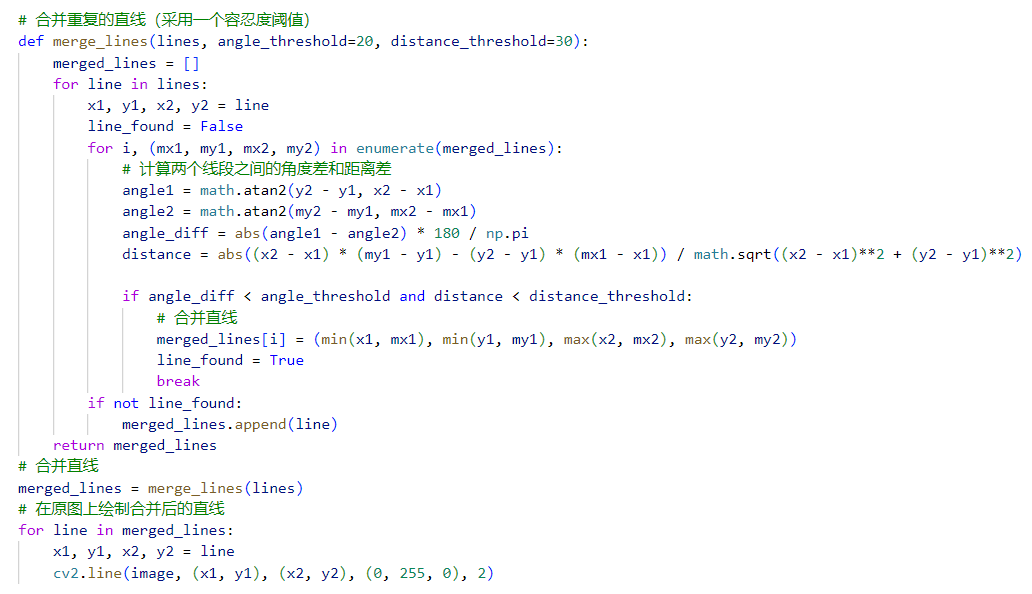


图3-4 合并直线代码

## 实验结果

本实验对校园道路图像进行了车道线检测，通过多个处理步骤获得了良好的检测效果。实验结果如图4-1所示，包含了从边缘检测到最终车道线标注的完整处理过程。

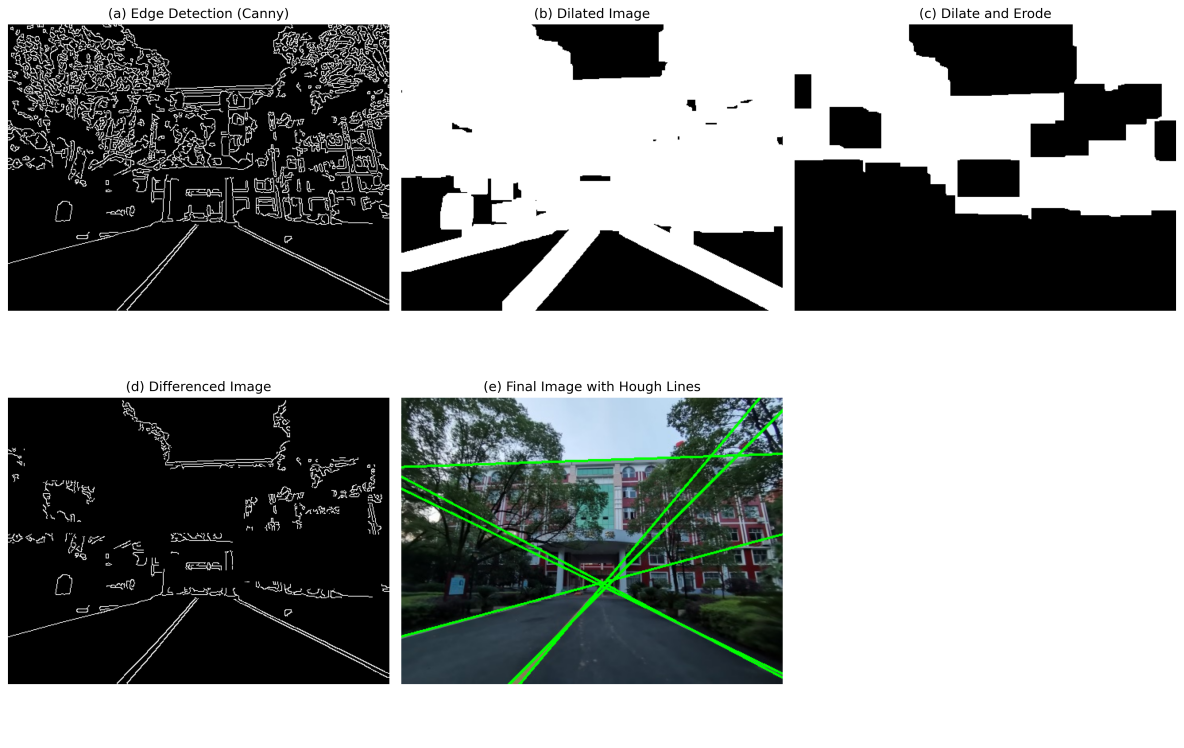


图4-1 直线检测各步骤结果

从图4-1(a)的边缘检测结果可以观察到，Canny边缘检测器成功提取出了图像中的主要边缘特征，道路的车道线边缘被清晰地检测出来，边缘连续性较好。然而，图像中也包含了大量环境噪声，如树木、建筑物的边缘等干扰信息。由于光照和阴影的影响，部分车道线的边缘存在断裂现象，这促使我们采用后续的形态学处理来改善检测效果。

形态学处理和差异化处理是本实验中的关键步骤。形态学处理采用膨胀和腐蚀操作（图4-1(b)和(c)），其原理是利用结构元素对图像进行局部特征提取。膨胀操作通过扩展边缘区域，能够连接由于光照不均、阴影或噪声导致的边缘断裂，这对于保持车道线的连续性至关重要。腐蚀操作则通过收缩区域来去除细小的噪声点，保留主要的结构特征。如果缺少膨胀腐蚀步骤，大量的环境噪声将会干扰霍夫变换，产生大量虚假的直线检测结果，同时断裂的边缘将导致后续霍夫变换无法正确检测完整的直线如图4-2所示。

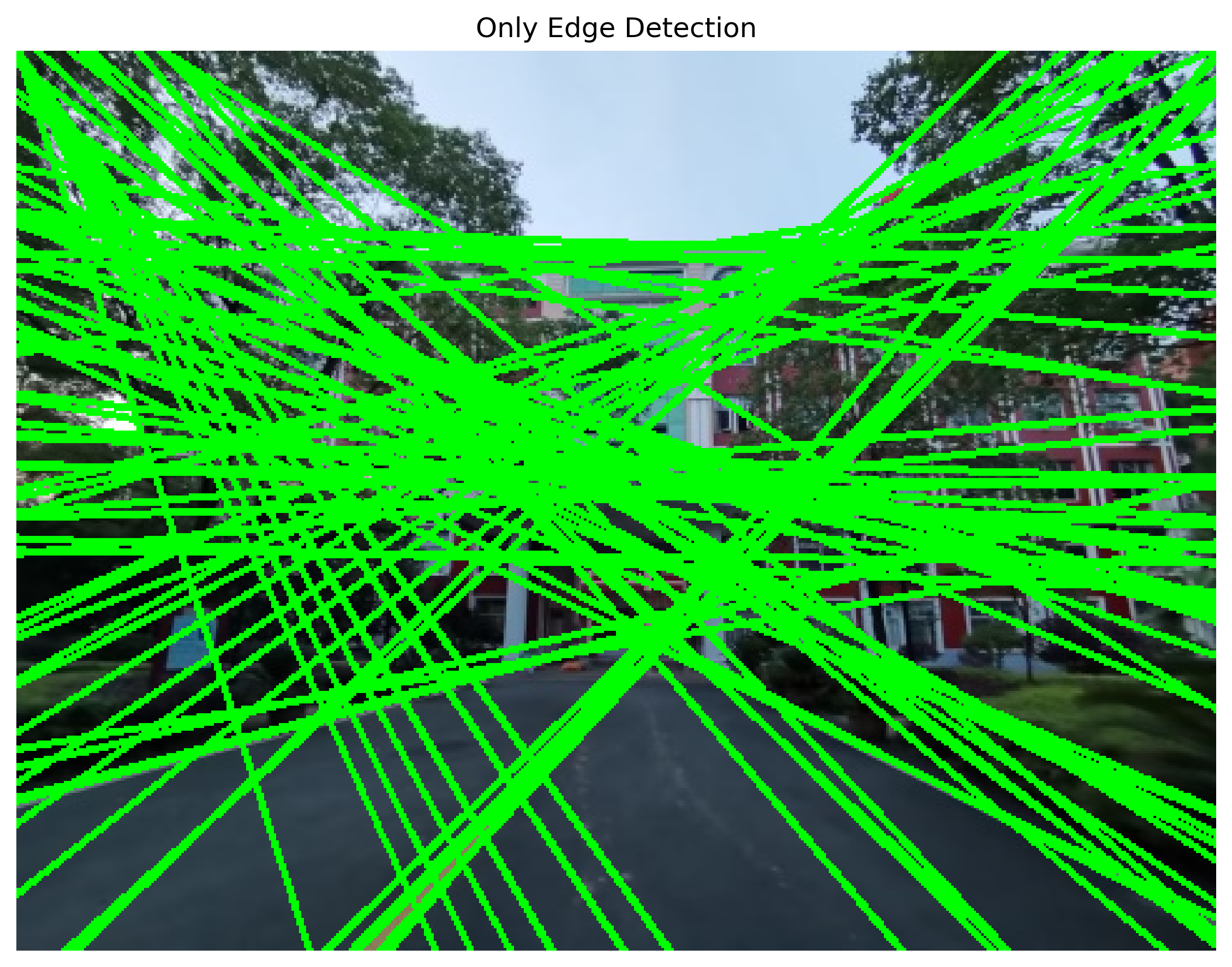


图4-2 不使用形态学操作的结果

差异化处理（图4-1(d)）是在形态学处理基础上的创新，其原理是利用原始边缘图像与腐蚀结果的差异来突出真实的车道线特征。这种处理方式特别有效，因为车道线通常具有较强的边缘响应，而噪声和背景特征在腐蚀后会显著减弱。通过差异化处理，我们可以进一步增强车道线的特征，同时抑制背景噪声。如果不进行差异化处理，直接使用形态学处理的结果，可能会保留过多的背景信息，导致霍夫变换在检测直线时出现大量误检如图4-3所示。

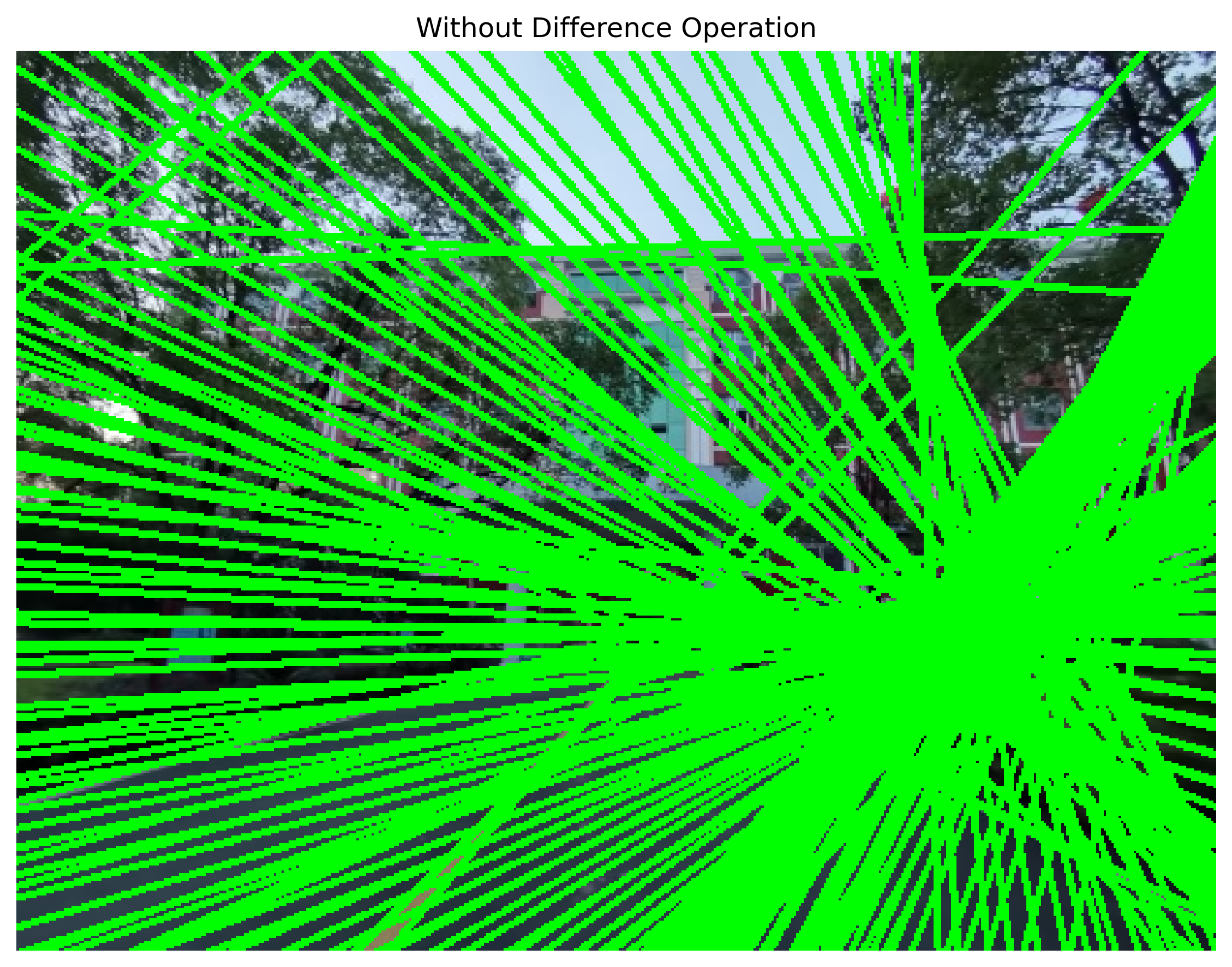


图4-3 不使用差异化操作

最终的检测结果如图4-1(e)所示，通过霍夫变换检测并经过筛选合并后的车道线（以绿色线条标注）准确地标示出了道路的边界。即使在存在树木阴影和复杂背景的情况下，算法仍能保持稳定的检测效果。检测出的车道线与实际道路标线具有很高的吻合度，没有出现重复检测的情况，这证明了形态学处理和差异化处理的有效性。特别是在处理复杂场景时，这些步骤的组合显著提高了算法的鲁棒性，使车道线检测更加准确和可靠。

## 实验体会

通过本次车道线检测实验，我深入理解了图像处理中从基础预处理到特征提取的完整流程，特别是在实现过程中体会到了算法选择和参数调优的重要性。在实验初期，仅使用简单的边缘检测就发现结果并不理想，这促使我思考如何通过多个处理步骤的组合来提升检测效果。

在实现形态学处理时，我认识到了膨胀和腐蚀这些基本操作在实际应用中对于二值图像噪声的巨大作用，它们不仅能够有效地处理图像噪声，还能弥补由于光照不均导致的边缘断裂问题。通过观察形态学处理的结果，发现在膨胀腐蚀后的图像与原边缘检测的二值图像相比，在原图具有大量噪声的地方经过形态学处理后转为了白色的一整块连通区域，而相对较细的直线部分则是转为了背景，所以利用原始边缘图像与膨胀腐蚀结果的差异可以更好地突出车道线特征并消除了大量的噪声，虽然图像中仍是有部分噪声，但是在后续可以通过对投票结果进行阈值化处理的方式筛除这些点可能产生的直线。在实现霍夫变换时，我进一步理解了参数空间转换的数学原理，以及如何通过累加器来检测直线特征。特别是在调整霍夫变换的阈值参数时，需要在检测灵敏度和抗噪声能力之间找到平衡点。但是实验不足的一点是无法消除天空与地平线的影响，由于这两部分的边缘的像素梯度通常很大，所以通过阈值的方式无法去除，若设置较大的阈值，则会将车道线一并删除了，后续可以采用深度学习的方式进行消除天空与地平线的交线带来的误差。

实验过程中遇到的各种问题也给了我很大启发。例如，在处理复杂背景时，单一的处理方法往往难以应对，需要综合运用多种技术才能获得理想效果以及需要通过观察中间结果来分析问题所在，并据此调整处理策略。这种调试和优化的经验对于今后解决类似的计算机视觉问题很有帮助。通过这次实验，我不仅掌握了车道线检测的技术方法，更重要的是理解了图像处理中"由简到繁"的设计思路。从最基础的边缘检测，到复杂的形态学处理和霍夫变换，每一步都是为了解决特定的问题，最终构建出一个完整的解决方案。这种渐进式的问题解决方法对于处理其他计算机视觉任务也具有重要的参考价值。

这次实验不仅提升了我的编程技能，也加深了我对图像处理理论的理解。在实现各种算法的过程中，我体会到了理论与实践相结合的重要性，以及如何在实际应用中灵活运用所学知识。这些经验对于今后继续深入学习计算机视觉和图像处理技术具有重要的指导意义。

# 实验三：学号识别

## 实验目的

* 掌握深度学习在计算机视觉领域中的基本应用
* 掌握深度学习环境搭建配置
* 理解并实现基于DenseNet的手写数字识别系统
* 学习图像预处理技术，包括二值化、轮廓检测等基本操作
* 掌握深度学习模型的训练和部署流程
* 实现从单个数字识别到连续数字（学号）识别的应用扩展

## 实验原理

### **2.1**深度学习环境配置

本实验基于Python环境进行开发，主要依赖PyTorch深度学习框架。PyTorch是一个开源的机器学习库，提供了灵活的动态计算图和丰富的深度学习工具。环境配置采用Anaconda进行Python包管理，主要安装以下关键组件：PyTorch作为深度学习框架的核心，torchvision提供计算机视觉相关的数据集和模型，OpenCV用于图像处理，以及用于数据分析和可视化的numpy和matplotlib等库。

### **2.2**数据集介绍

MNIST数据集是一个经典手写数字数据集，包含60,000张训练图像和10,000张测试图像，数据集分布与样例如图2-1所示。每张图像都是28×28像素的灰度图，像素值范围在0-255之间。数据集中的每个数字都经过归一化和中心对齐处理，这使得MNIST成为深度学习入门的理想数据集。数据集中的图像具有以下特点：背景为黑色（像素值为0），数字为白色（像素值接近255），图像大小统一，这些特点有助于模型学习数字的本质特征。

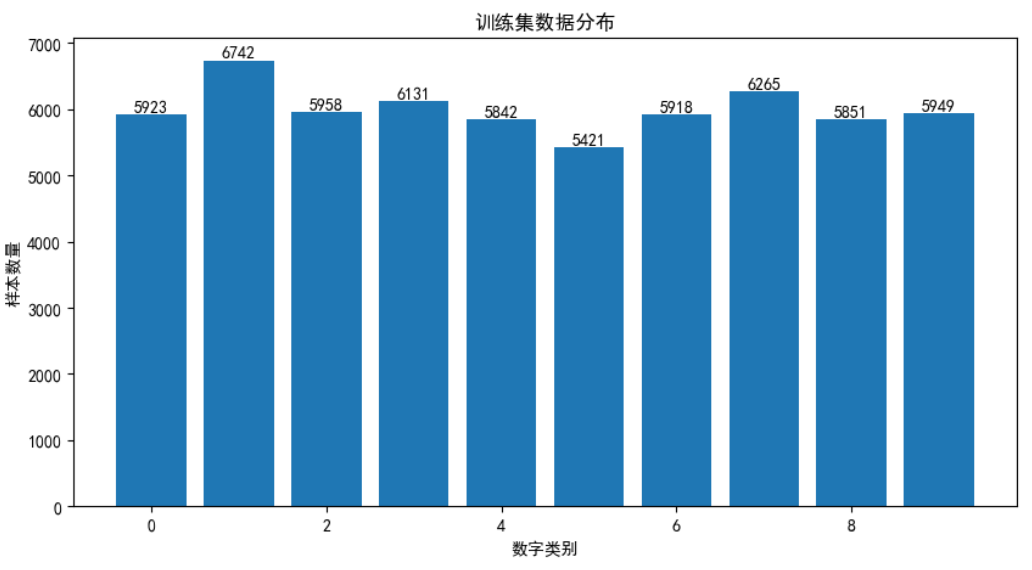
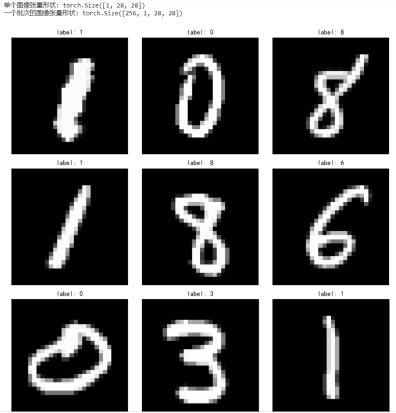


图2-1 数据集示例

### **2.3**DenseNet网络

DenseNet（密集连接网络）是一种高效的深度学习架构，通过建立层与层之间的密集连接来提高特征利用率和梯度流动。在传统的卷积神经网络中，每一层仅接收前一层的输出作为输入，而DenseNet中每一层都会接收其前面所有层的特征图作为输入，形成密集连接模式。如果将第l层的输出表示为xl，该层的输入将是前面所有层的特征图的组合：，其中表示将前面所有层的特征图在通道维度上进行连接，是一个非线性转换函数。

在本实验中实现的DenseNet结构如图2-2所示，每个Dense Layer包含批归一化、ReLU激活函数和3×3卷积操作。growth\_rate参数控制着每层产生的新特征图数量，这个参数决定了网络增长的速度。Dense Block是DenseNet的核心组件，其中的每一层都与后续所有层直接相连。如果一个Dense Block包含n层，那么将有个直接连接。这种密集连接的设计不仅加强了特征的重用，也显著减少了需要学习的参数数量。

在Dense Block之间，使用Transition Layer来控制特征图的维度。Transition Layer包含1×1卷积和2×2平均池化操作，用于降低特征图的空间维度和通道数。这种设计既保持了网络的特征表达能力，又控制了计算复杂度。

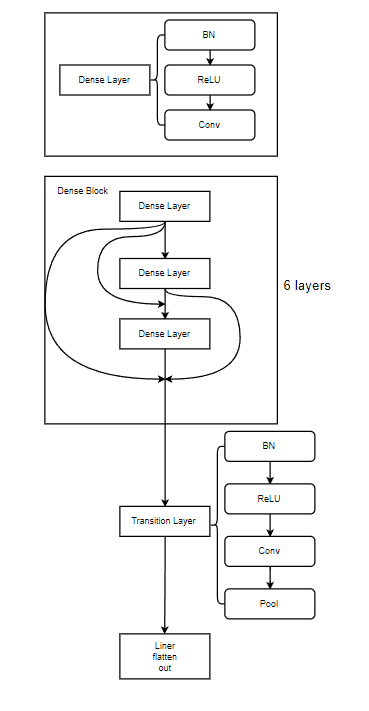


图2-2 本实验DenseNet架构

在特征传播过程中，Dense Block内的每一层都会产生个特征图（为growth rate），这些特征图会被连接到后续层的输入中。假设输入层有个特征图，那么第l层的输入特征图数量将是。这种连接方式使得网络能够最大限度地利用所有层的特征，同时也有助于减轻梯度消失问题。通过反向传播，每一层都能直接获得来自损失函数的梯度，这大大提高了网络的训练效率。

在本实验的手写数字识别任务中，我们使用了较小规模的DenseNet结构，包含4个Dense Block，每个Block包含6层，growth rate设置为12。初始卷积层使用64个3×3的卷积核，最后通过全局池化和全连接层将特征映射到10个类别上。这种结构在保持较低参数量的同时，能够有效地提取手写数字的特征，实现高精度的识别。

### **2.4**图像预处理

在输入的图像预处理中，需要使用一种通用的方式将图像尽可能都转换为和训练图像类似的样式。对于学号图像的处理，首先将彩色图像转换为灰度图，然后通过二值化处理突出数字轮廓。使用OpenCV的轮廓检测算法找到每个数字的位置，并按照从左到右的顺序对数字进行排序。对于每个检测到的数字区域，将其裁剪并缩放到28×28像素大小，与MNIST数据集的格式保持一致。最后对图像进行归一化处理，将像素值缩放到[-1,1]区间，这样可以进一步增加模型部署使用的准确性。

## 实验方法

本实验构建了一个基于DenseNet的手写数字识别系统，从环境配置到模型部署形成完整的工作流程。首先基于Anaconda配置Python深度学习环境，安装必要的依赖包。然后使用PyTorch框架搭建DenseNet模型，通过MNIST数据集进行模型训练。最后使用Streamlit框架开发Web交互界面，实现模型的实际部署应用。整个实验流程形成了一个闭环的工作链，确保了从开发到应用的顺畅过渡。

### **3.1**深度学习环境配置

在实验开始前，需要进行深度学习环境的搭建。本实验采用的是Anaconda构建环境，需要在Anaconda官网下载相应安装包并安装，在完成后能够启动Anaconda Prompt，后续创建虚拟环境在可以该终端下进行（或者在git bash中亦可）。

打开Anaconda Prompt，输入conda create -n hsx python=3.11即可创建一个名为hsx，python版本为3.11的虚拟环境，这种方式带来的好处是能够一定程度上减少库版本冲突的可能性，同时能够在发生冲突时重新删除并创建一个新的环境而不影响主机的环境。随后可以使用conda env list查看当前创建了哪几个环境如图3-2所示。

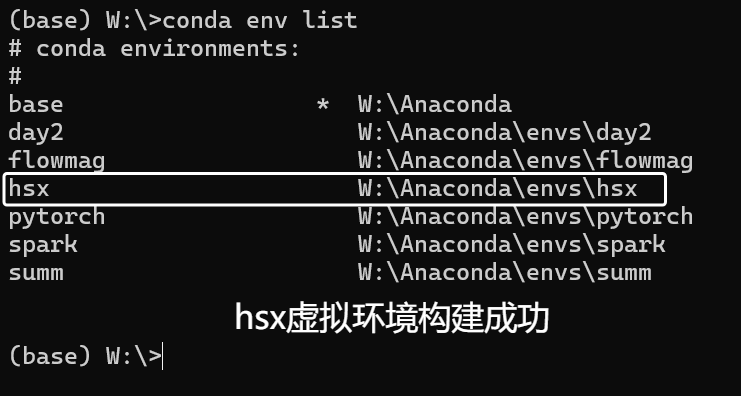
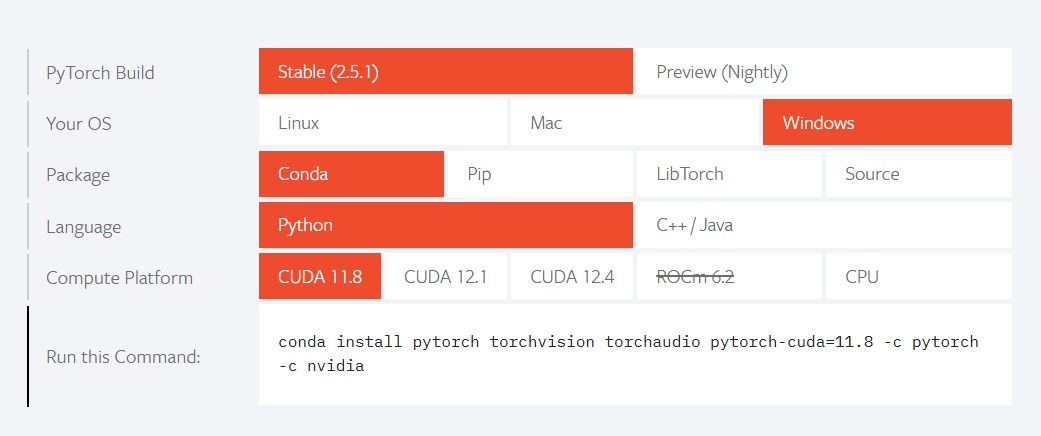


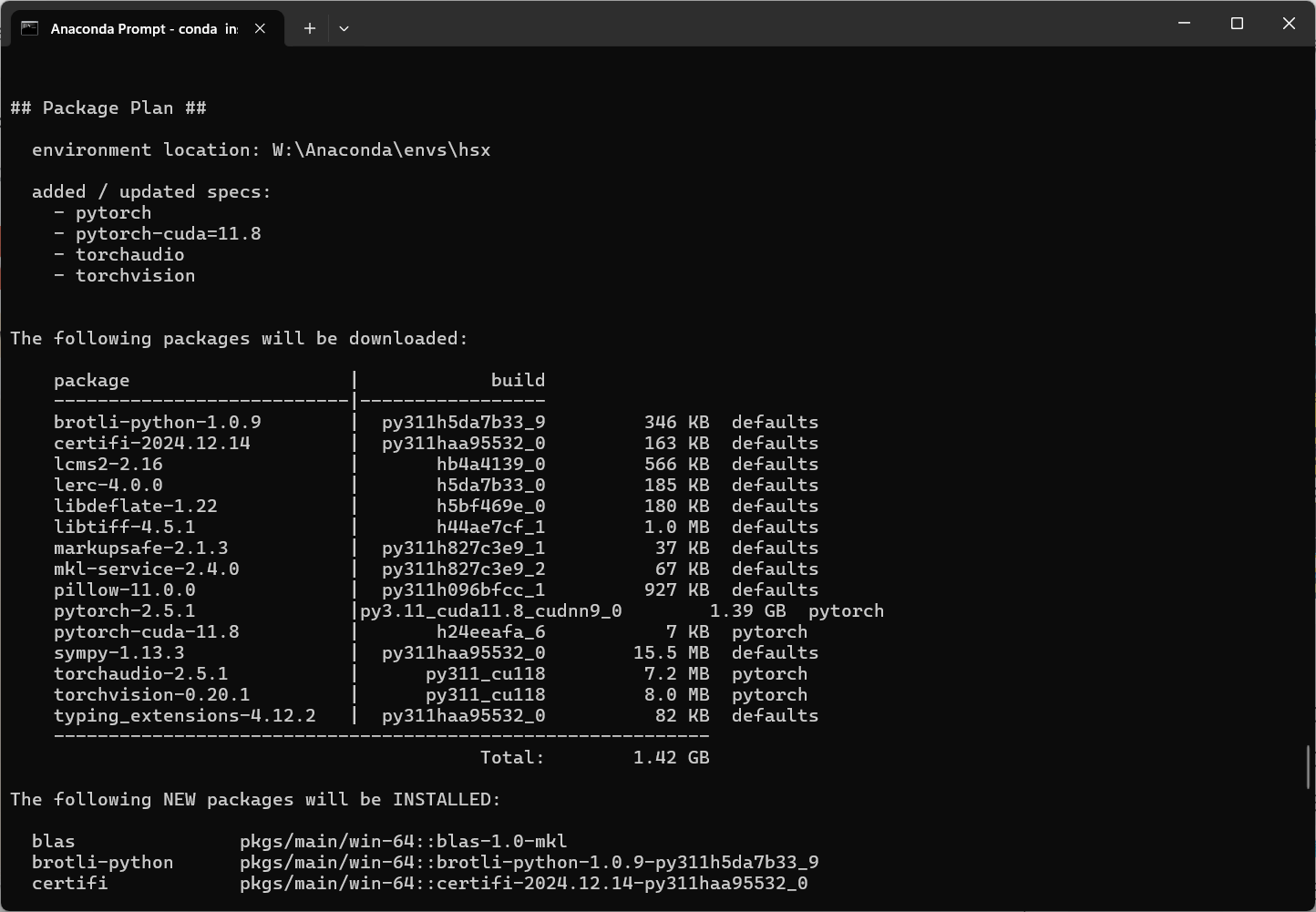
图3-1 查看是否构建成功虚拟环境

在成功创建虚拟环境后使用conda activate hsx进入hsx虚拟环境并提前使用nvidia-smi查看本机对应的CUDA版本，随后开始安装pytorch深度学习库。

打开pytorch官网如图3-3所示，由于我的电脑CUDA版本为12.3，为了稳妥起见，我选择了CUDA=11.8的pytorch版本进行安装，复制下方的命令到Anaconda Prompt中即可（网络不好可能在Solving environment阶段需要等一段时间才会提示输入y开始安装）。



1. pytorch官网选择



(b) 安装过程

图3-2 安装pytorch

安装好后，在终端启动python并输入import torch、torch.\_\_version\_\_、touch.cuda.is\_available()分别查看pytorch版本和自己电脑上的GPU是否能使用，如图3-4所示，随后便可在hsx这个环境下安装opencv\_python等其它库。

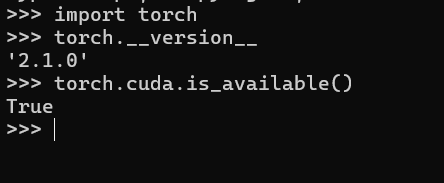


图3-3 查看是否安装成功

### **3.2**DenseNet模型搭建

DenseNet模型的搭建采用PyTorch框架，模型结构包含初始卷积层、多个Dense Block和Transition Layer。初始卷积层使用64个3×3的卷积核对输入图像进行特征提取。每个Dense Block包含6层密集连接的DenseLayer，growth rate设置为12，用于控制特征图的增长速度。Dense Block之间通过Transition Layer进行特征图的维度调整，包含1×1卷积和2×2平均池化操作。在每个DenseLayer内部，采用批归一化（Batch Normalization）、ReLU激活函数和卷积层的组合进行特征提取。这种密集连接的结构设计使得网络能够充分利用浅层特征，同时通过特征重用减少了参数数量，提高了计算效率。Transition Layer的设计则有效控制了特征图的维度，防止网络参数过度增长。最后通过全局平均池化和全连接层将特征映射到10个类别的输出空间，完成手写数字的分类任务。

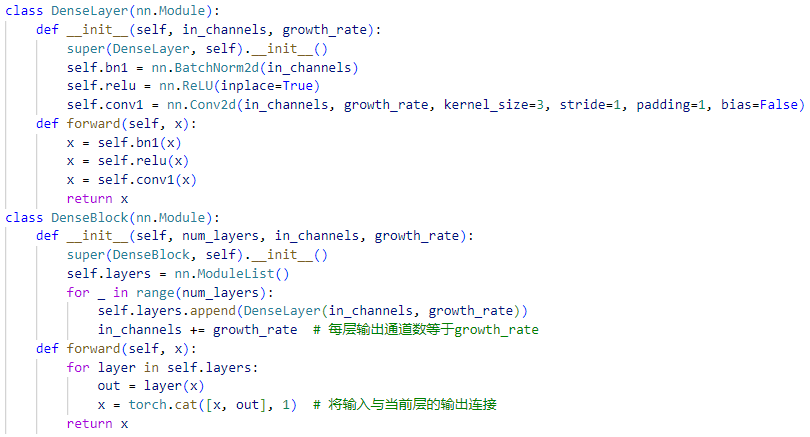




图3-4 DenseNet搭建代码

### **3.3**数据集导入

数据集处理使用torchvision库提供的MNIST数据集接口。在数据导入过程中，首先通过ToTensor转换将图像数据转换为PyTorch张量格式，并使用Normalize进行数据归一化处理，将像素值缩放到[-1,1]区间。训练集包含60,000张图像，测试集包含10,000张图像，所有图像都是28×28像素的灰度图。数据加载过程中采用DataLoader进行批处理，设置批次大小为256，并在训练集上启用随机打乱以增强模型的泛化能力。同时，实现了数据增强策略，包括随机旋转和平移，以提高模型对各种手写风格的适应能力。

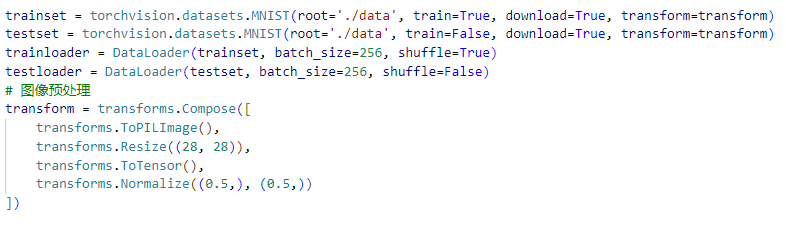


图3-5 数据集导入部分

### **3.4**模型训练

模型训练采用Adam优化器，初始学习率设置为0.001，并使用学习率衰减策略，每10个epoch将学习率降低为原来的0.1倍。考虑到这是多分类问题的标准选择，所以损失函数使用交叉熵损失。训练过程在GPU上进行，总共进行了50轮训练。在每轮训练中，模型依次处理所有批次的数据，计算预测结果与真实标签之间的损失，并通过反向传播更新模型参数。为了防止过拟合，实现了早停策略，当验证集上的损失连续5个epoch没有改善时，提前终止训练。同时，使用模型检查点机制，保存训练过程中性能最好的模型参数。

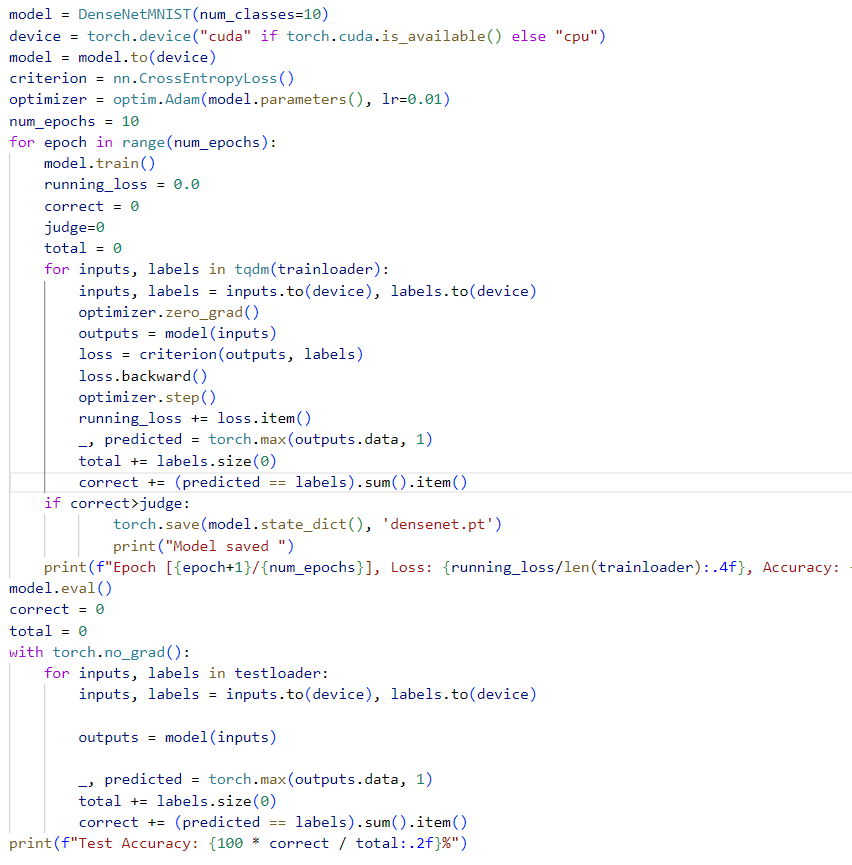


图3-6 模型训练部分

### **3.5**模型前端部署

模型的前端部署使用Streamlit框架构建交互式Web界面。界面设计包含两个主要功能标签页：单个数字识别和学号识别。图像处理流程采用多阶段预处理策略，确保输入图像能够被模型准确识别。

在图像预处理阶段，系统首先将RGB图像转换为灰度图以减少计算复杂度并使用高斯滤波器去除图像噪声提高识别稳定性。接着采用自适应阈值算法进行二值化处理，这种方法相比全局阈值更能适应不同光照条件下的图像。随后通过膨胀操作去除小的噪点。

在数字分割阶段，系统使用OpenCV的findContours函数检测图像中的所有轮廓，并基于面积、宽高比等特征筛选出有效的数字轮廓。为确保准确识别学号中的连续数字，进行基于轮廓中心点x坐标的排序从而保证数字按从左到右的顺序识别。对于每个分割出的数字区域，系统进行了边界框计算，并在提取时添加适当的边距，避免数字被误截断，同时，在测试时发现仅仅是使用这些操作，裁剪出的数字图像通常是最小化边界框即图片长宽无线逼近于数字的长宽。由于每个图片的边界框都挨着数字区域所以图片的宽度通常会小于20像素，但是后续需要将图片调整为28×28的大小，所以在进行调整后数据已经失去了其原本的特征，如图3-7(a)所示。所以为了避免这个情况我将每一张图片的外围填充背景像素（即黑色像素），由于每个图片都是长＞宽，所以在每一张分割出来的数字图中的上下分别添加5个背景像素，左右添加9个像素，这样转换出来的图片如图3-7(b)所示，这样处理后能够很好的解决上述的问题，也对后续的识别有突出的提升作用。

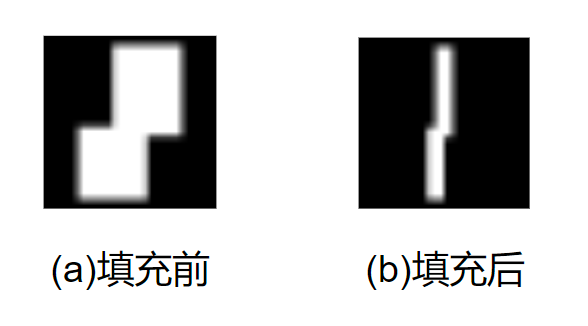


图3-7 填充前后的对比图（图为进行28×28放大后的结果）

处理完上述步骤后，系统将每个分割出的数字调整为28×28像素大小，与MNIST数据集保持一致。同时进行像素值归一化，将值域缩放到[-1,1]区间，确保与模型训练时的数据分布匹配。在调整过程中，系统会保持数字的宽高比，必要时通过边界填充保持数字在图像中的居中位置。这种标准化处理确保了输入数据的一致性，提高了模型识别的准确率。界面还实现了实时预览功能，用户可以看到图像处理的中间结果，包括二值化效果、分割结果等。为了提高用户体验，添加了批量处理功能，支持同时识别多张图片，并提供了识别结果的导出功能。系统集成了错误处理机制，能够优雅地处理各种异常情况，如图像格式不正确、识别失败等。通过这种完整的图像处理流程，系统能够有效处理各种不同条件下拍摄的手写数字图像，实现稳定可靠的识别效果。



图3-8 前端构建核心代码

## 实验结果

模型在MNIST数据集上的训练过程表现出优异的性能。经过10轮训练，损失函数呈现稳定下降趋势并达到了0.017，没有出现明显的震荡或过拟合现象。最终模型在训练集上达到99.52%的准确率，在测试集上达到99.33%的准确率，表明模型具有良好的泛化能力，由于网络学习能力很好，所以只训练了10轮以免过拟合的发生；同时，单样本推理时间平均为0.16秒，一定程度上也满足了实时检测的需求。训练过程中的损失函数和准确率变化趋势如下图4-1所示。

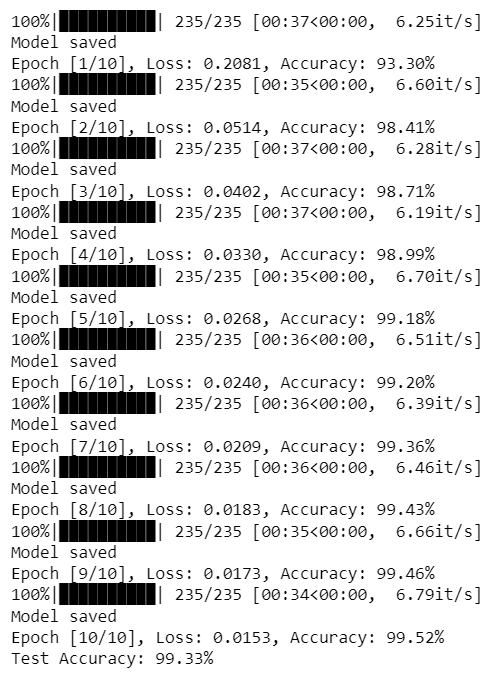


图4-1 模型训练结果

使用streamlit设计的系统前端界面实现了单个数字识别和学号识别两个主要功能，在单个数字识别界面中，系统清晰展示了图像处理的完整流程，包括原始图像、二值化结果、裁剪后的图像以及最终的识别结果如图4-2所示。可以点击选择一张手写数字上传进行识别，能够直观地观察到图像从输入到识别的整个过程。

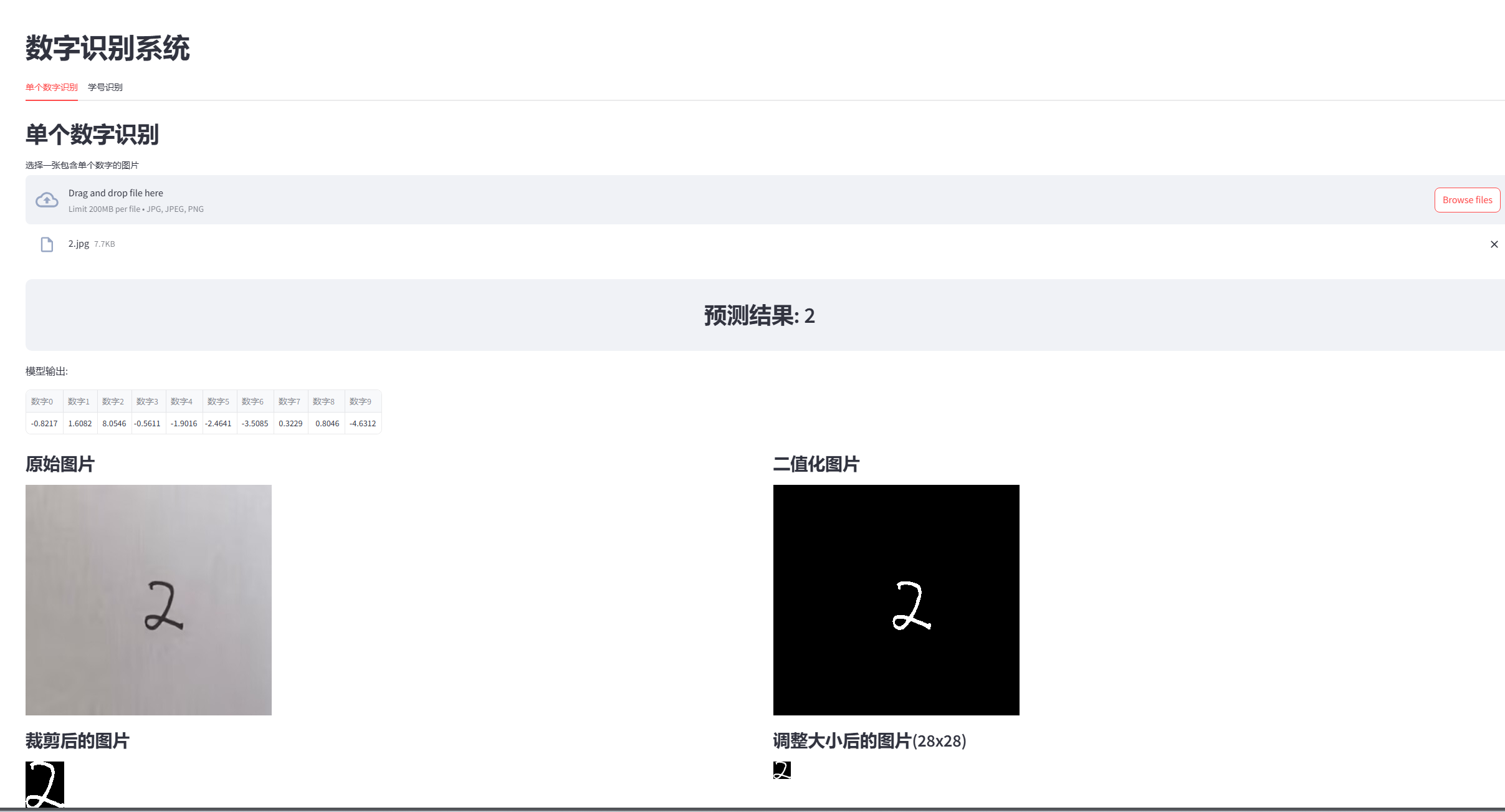


图4-2 前端界面展示

对于学号识别功能，系统展示了数字分割的过程，即展示了各步骤处理后的图像变化。同时，每个分割出的数字都标注了边界框和识别结果，并最终合成了完整的学号序列输出在下方如图4-3所示。

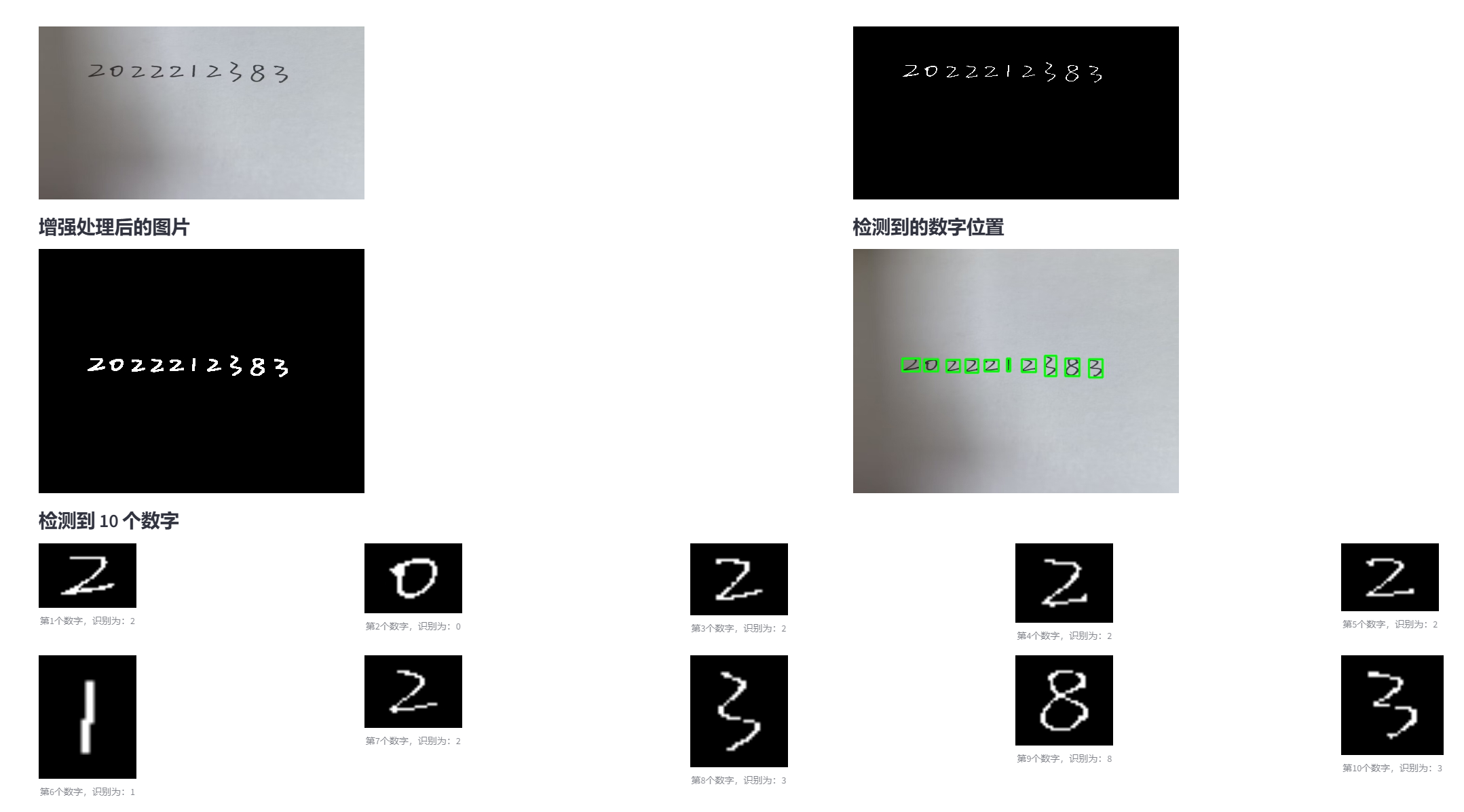
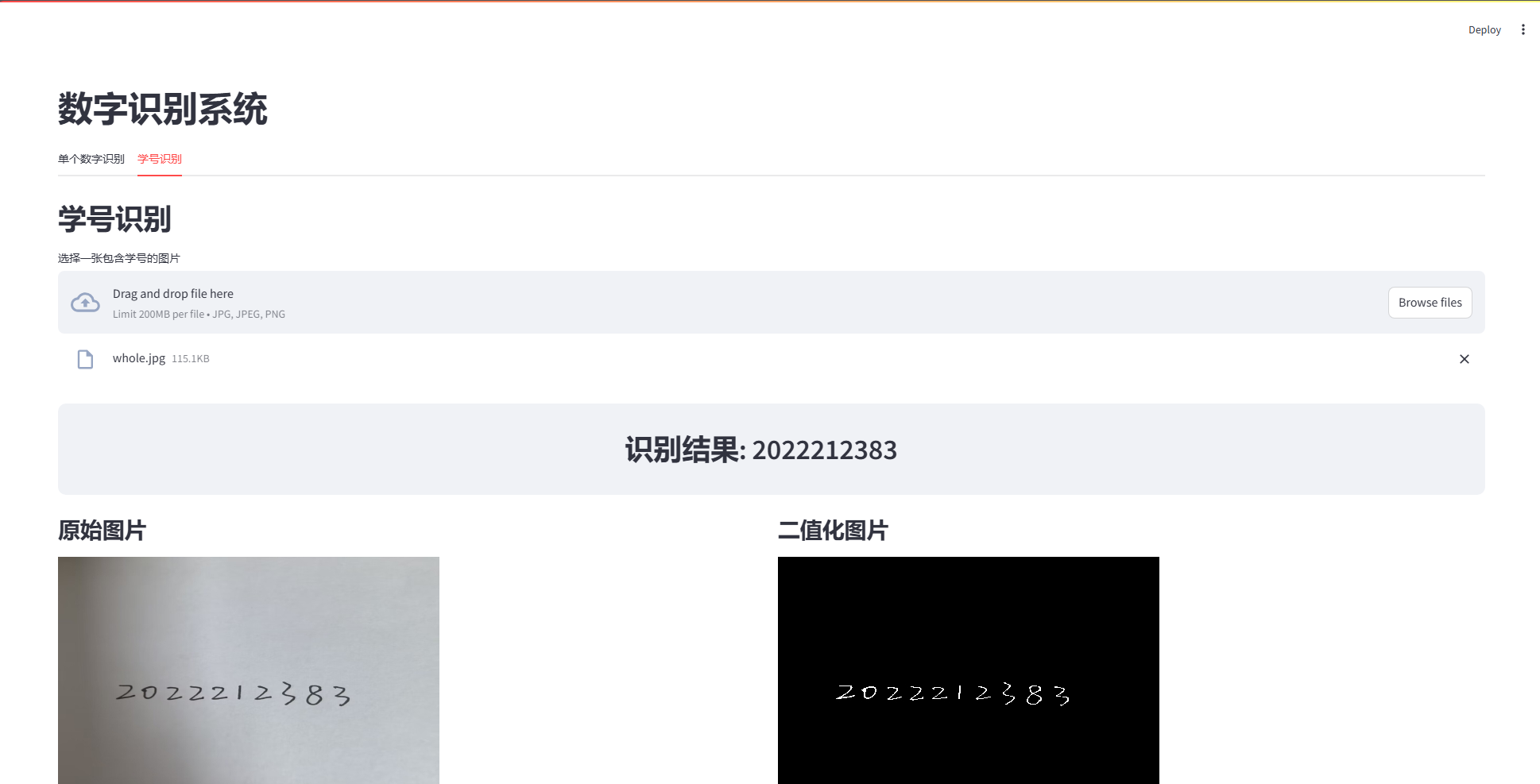


图4-3 学号识别展示

## 实验体会

通过本次基于DenseNet的手写数字识别系统的设计与实现，我深入理解了深度学习在计算机视觉领域的实际应用流程。从环境搭建、模型构建到系统部署的每个环节，都让我对深度学习项目的开发流程有了更加系统的认识。在实验过程中，我深刻体会到理论知识与实践操作的紧密联系，以及工程实现中可能遇到的各种挑战和解决方案。

在模型设计阶段，我初步使用了LeNet-5模型架构对于MNIST数据集进行训练，但是训练结果是在训练集上的准确率很高，但是在测试集和验证集上的准确率低同时实际使用准确性也不高，初步考虑是网络层太简单导致特征提取不足，无法实现泛用性，所以后续使用了网络层数较多的DenseNet，该网络结构不仅能够充分利用特征信息，还能有效缓解梯度消失问题，提高模型的训练效率。通过实验发现，合理的网络结构设计对模型性能的提升起着关键作用。同时，在调试模型参数的过程中，我深刻理解到超参数调整的重要性，如何平衡模型的复杂度和性能成为了一个重要的考虑因素。

在数据处理环节，我认识到数据质量对模型训练的重要影响。通过设计合适的数据增强策略，不仅提高了模型的泛化能力，还使模型在面对不同书写风格时表现出更好的鲁棒性。特别是在处理实际图像时，我发现仅仅依靠基本的图像处理方法往往无法获得理想的效果。例如，在数字分割过程中，最初使用简单的轮廓提取会导致裁剪出的数字图像过于紧凑，当将其调整为标准大小时会严重失真。通过添加适当的边界填充（上下各5像素，左右各9像素的黑色背景），不仅保持了数字的原始比例，还显著提升了识别准确率。这些经验让我明白，在实际应用中，输入测试的数据需要符合训练的数据的样式才能进一步提高使用的准确性。

在系统部署阶段，我学会了使用Streamlit框架构建前端界面。通过不断优化图像处理流程，如实现自适应阈值算法进行二值化处理，使系统能够更好地适应不同光照条件下的图像。在数字分割阶段，通过基于轮廓中心点x坐标的排序算法，确保了连续数字的正确识别顺序。这些细节的优化最终提升了系统的整体性能和用户体验。在图像预处理环节，通过多次实验和改进，最终找到了一套有效的处理流程：从RGB转灰度、高斯滤波去噪、自适应阈值二值化到合理的边界填充，每一步的优化都对最终的识别效果产生了显著影响。

通过这次实验，我不仅掌握了深度学习的基本原理和实现方法，更重要的是学会了如何将深度学习技术应用到实际问题中。这次实验也让我意识到，人工智能技术的发展正在改变着我们的生活方式，我们需要不断学习和创新，将先进的技术转化为解决实际问题的有效工具。

# 附录：关于通过Github提交的方式

在使用github前，我们需要创建一个github账号以及在自己电脑上安装git bash软件用来与github进行互联。

安装好git后，打开git bash命令行，输入git config --local user.name 和git config --local user.email命令设置本机的信息，并且生成SSH公钥复制到github个人账户中，如图1所示。添加完成后可以输入ssh -T git@github.com命令验证是否添加成功。

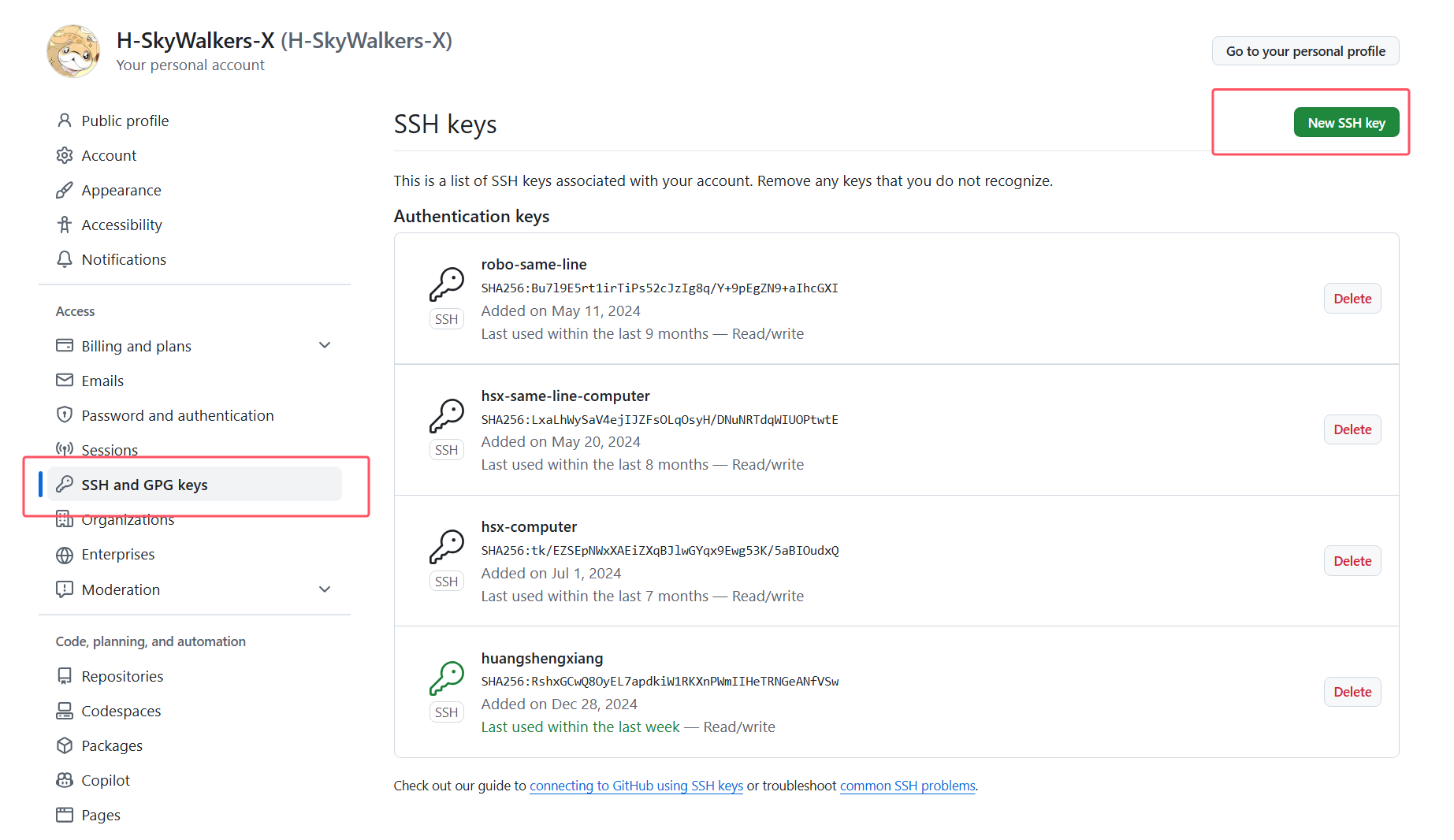


图1 复制SSH公钥

添加成功后，在github主页上创建一个新项目并在项目主页中复制该项目的路径如图2所示，并在本机使用git clone+路径的命令将项目拉取到本地，随后在这个项目文件夹中的修改可以直接push到github仓库中。

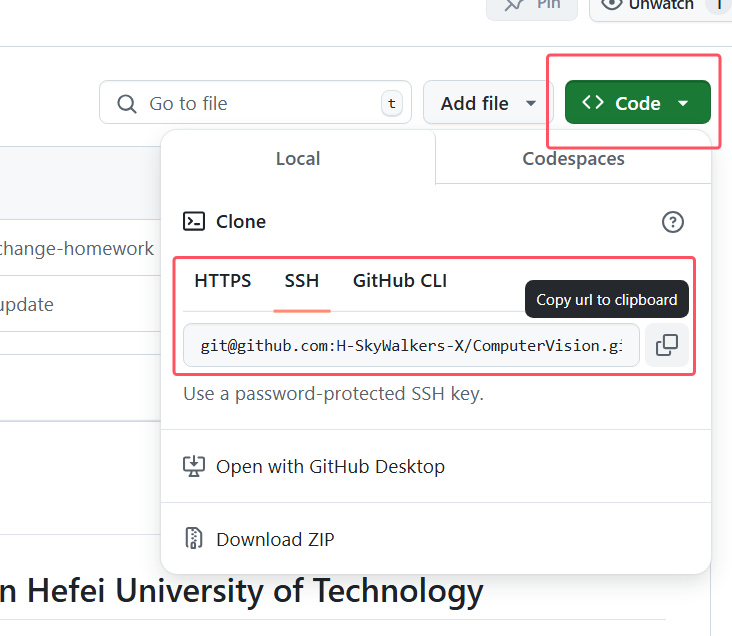


图2 复制该项目的路径

将需要上传的文件放到clone到本地的文件夹中并使用git add .将所有的文件（修改的文件）提交到本地仓库的提交缓存并使用git commit -m “描述信息”提交到本地仓库，此时会输出修改的信息，如图3所示。

图3 提交到本地仓库的输出信息