数字图像处理
第四次作业

项目名称：数字图像处理第四次作业

班级：　　自动化 2104

姓名：　　马茂原

学号：　　2216113438

提交时间：2024 年 3 月 16 日

摘要：

本实验旨在探索图像处理中常用的空域滤波技术,包括低通滤波和高通滤波。低通滤波部分采用高斯滤波器和中值滤波器对测试图像 test1 和 test2 进行平滑处理,模板大小分别为 3x3、5x5 和 7x7。实验分析了不同滤波器和模板大小对图像平滑效果的影响,讨论了各自的优缺点。此外,利用固定方差 σ =1.5 生成高斯滤波器,探讨了方差选择对滤波效果的影响。

高通滤波部分应用了多种边缘检测算法,包括 Unsharp masking、Sobel 边缘检测器、Laplace 边缘检测和 Canny 算法,对测试图像 test3 和 test4 进行边缘提取。实验对比了不同算法的边缘检测效果,分析了各算法的优缺点及适用场景。

通过对低通滤波和高通滤波的实践探索,本实验理解了空域滤波在图像处理中的重要作用。实验结果有助于掌握这些滤波技术的使用方法,并为后续的图像处理任务提供有价值的参考。

关键字：空域低通滤波器，高通滤波器

题目一. 空域低通滤波器：分别用高斯滤波器和中值滤波器去平滑测试图像 test1 和 2，模板大小分别是 3x3 ， 5x5 ，7x7； 分析各自优缺点；

1. 技术分析

A . 高斯滤波器

高斯滤波器[1](Gaussian Filter)是一种线性平滑滤波器,在图像处理中被广泛应用于减少图像噪声、模糊和图像平滑等。它的原理是用高斯函数计算的加权平均值来代替图像中每个像素点的值。高斯滤波器具有很好的去噪性能,并能很好地保留边缘细节。

高斯滤波器的工作原理如下:

1. 定义高斯核函数

高斯核函数其公式为:

$$\mathrm{k}_{i,j} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-d-1)^2+(j-d-1)^2}{2\sigma^2}}$$

其中σ是高斯核函数的标准差,用于控制滤波器的平滑程度。σ越大,卷积核尺寸越大,平滑效果越明显。

2. 构造高斯卷积核

通过对高斯核函数进行离散化和归一化,可以得到一个二维高斯卷积核矩阵,如 3×3、5×5 等。

3. 滤波操作

将高斯卷积核在整个图像上滑动,对每个像素点进行加权求和计算,得到该像素新的灰度值。

B. 中值滤波器

中值滤波器[2],常用于图像处理中去除噪声。它的基本原理是用像素点邻域灰度值的中值来代替该像素点的值,从而达到平滑图像的目的。

其工作步骤如下:

1. 选取窗口:首先以目标像素点为中心,选取一个包含有奇数个像素点的窗口区域,窗口的大小通常为 3×3、5×5 或更大。

2. 排序:将窗口内所有像素点的灰度值进行排序。

3. 取中值:取排序后像素值的中值。

4. 赋值:用步骤 3 得到的中值代替原始图像中相应像素点的灰度值。

5. 滑动窗口:对图像中的其他像素点重复上述步骤,直至扫描完整个图像。

2. 运行结果

使用高斯滤波器平滑 test1 和 test2 的结果如图 1 和图 3 所示，使用中值滤波器平滑 test1 和 test2 的结果如图 2 和图 4 所示。

图 1. test1 使用高斯滤波器平滑结果

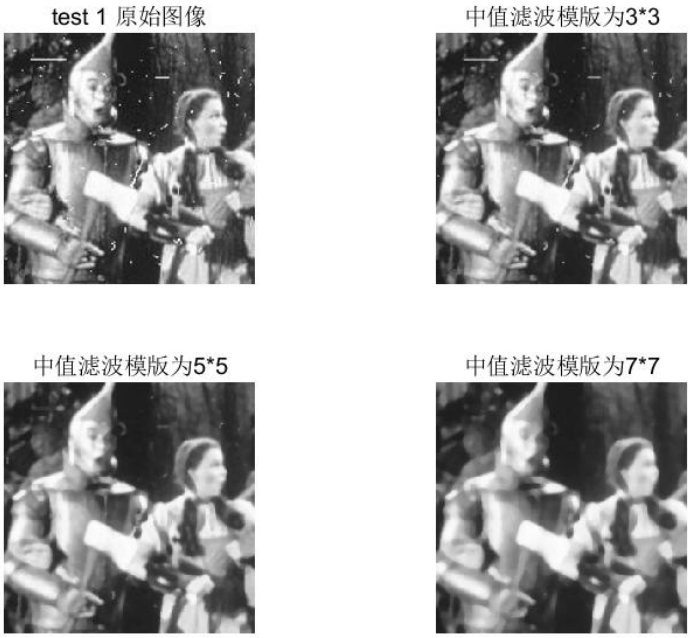

图 2. test1 使用中值滤波器平滑结果

图 3. test2 使用高斯滤波器平滑结果

图 4. test2 使用中值滤波器平滑结果

高斯滤波器具有良好的去噪性能，对图像中的高斯白噪声有很好的抑制作用，并且高斯核满足可分离性,可以将二维卷积分解为两个一维卷积,提高了计算效率。但是，高斯滤波器对脉冲噪声的抑制效果相对较差。且在σ选取较大时,会导致图像细节丢失而过度模糊。通过合理选择标准差σ,可以在降噪和保护细节之间达到平衡。

中值滤波器对椒盐噪声有良好的去噪效果,能在平滑噪声的同时很好地保留图像边缘细节。对噪声具有稳定性，对像素值的极端差值不太敏感,所以对异常值比较稳定。

由以上图像处理的结果可知，中值滤波的效果好于高斯滤波的效果。kernel的尺寸越大，滤波效果越明显，但图像的细节则会有所损失，需要平衡图像模糊程度与图像细节保留之间的关系。

题目二. 利用固定方差 sigma=1.5 产生高斯滤波器. 附件有产生高斯滤波器的方法； 分析各自优缺点；

1. 技术分析

实现高斯滤波器的方案如下：

1. **确定滤波器大小：**基于方差 sigma 确定的滤波器的大小，方差的大小决定了滤波器包含的范围，决定了滤波器包含的能量区间。

2. 创建滤波器矩阵:在编程中，需要创建一个二维数组来表示滤波器。数组的每个元素都是根据其到中心的距离计算出的高斯函数值。

3. **归一化滤波器**:为了保证滤波后的图像亮度不变，需要对滤波器矩阵进行归一化处理，确保所有元素的和为 1。

2. 运行结果

Python 实现高斯滤波器：

```
import numpy as np
import cv2

# 设定方差
sigma = 1.5
# 设定滤波器大小
filter_size = 6 * sigma + 1

# 创建高斯滤波器
x = np.linspace(-int(filter_size / 2), int(filter_size / 2), filter_size)
gauss_filter = np.exp(-x**2 / (2 * sigma**2))
gauss_filter = gauss_filter / gauss_filter.sum()

# 二维高斯滤波器
gauss_filter_2d = np.outer(gauss_filter, gauss_filter)

# 显示滤波器
cv2.imshow('Gaussian Filter', gauss_filter_2d)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

matlab 实现高斯滤波器：

A．调用自带函数

```
sigma = 1.5;
filter_size = 6 * sigma + 1;

% 创建高斯滤波器
hsize = [filter_size filter_size];
gauss_filter = fspecial('gaussian', hsize, sigma);

% 显示滤波器
figure, imshow(gauss_filter, []);
```

B．自己编写函数

```
function Img_out = GaussianFilter(Img, masksize, sigma)
    % 计算一维高斯核
    center = ceil(masksize/2);
    gauss1D = exp(-((-center+1:center-1).^2)/(2*sigma^2));
    gauss1D = gauss1D / sum(gauss1D); % 归一化

    % 通过外积得到二维高斯核
    h = gauss1D' * gauss1D;

    % 应用高斯滤波器
    Img_filtered = conv2(double(Img), h, 'same');

    % 归一化并转换为 uint8
    Img_out = uint8(Img_filtered / max(Img_filtered(:)) * 255);
end
```

自己实现函数具有灵活性，可以根据特定需求进行定制，例如添加特殊的边界处理或优化算法。但是，自定义函数可能在执行效率上不如专门优化过的内置函数。使用内置函数通常经过优化，能够提供更快的执行速度。经过广泛测试，错误风险较低。

题目三. 利用高通滤波器滤波测试图像 test3,4：包括 unsharp masking, Sobel edge detector, and Laplace edge detection；Canny algorithm.分析各自优缺点；

1. 技术分析

A．unsharp masking

unsharp masking 是图像处理中一种常用的锐化增强技术[3],目的是增强图像的细节和边缘对比度,使图像看起来更加清晰锐利。它的原理是通过计算图像与其模糊版本之间的差值,然后对该差值进行放大合成,从而增强图像的边缘和细节部分。具体步骤如下:

1. 模糊原始图像

首先需要对原始输入图像进行模糊处理,通常采用高斯滤波器或均值滤波器等滤波操作,生成一个平滑版本的模糊图像。

2. 计算高频残差

将模糊图像从原始图像中减去,得到一个高频残差图像,该图像突出了原始图像中的边缘、细节等高频部分。

残差 = 原始图像 - 模糊图像

3. 放大高频残差

为了增强边缘和细节,需要对高频残余进行放大处理,通常会乘以一个大于 1 的常数因子 k。

放大残差 = k * 残差

4. 合成锐化图像

最后将放大后的高频残差与原始图像相加,得到最终的锐化输出图像。

锐化图像 = 原始图像 + 放大残差

B．Sobel 边缘检测

Sobel 边缘检测器是一种常用的边缘检测算法[4],它利用一对 3x3 的卷积核,对图像进行离散微分运算,从而检测出图像中的水平、垂直和对角边缘。

其工作原理如下:

1. 构造 Sobel 卷积核

Sobel 算子包含两个 3x3 的卷积核,一个用于检测水平方向的梯度分量(Gx),另一个用于检测垂直方向的梯度分量(Gy)。

2. 计算梯度近似值

将上述两个卷积核分别与原始图像进行卷积运算,得到水平和垂直两个方向上的梯度近似值 Gx 和 Gy。

3. 计算梯度幅值和梯度方向

利用这两个梯度分量,计算每个像素点的梯度幅值 G 和梯度方向θ，梯度幅值 G 越大,说明该点越可能是边缘点。梯度方向θ用于确定边缘的方位。

C．Laplace 边缘检测

Laplace 边缘检测是一种基于二阶导数的边缘检测算法[5],它对图像进行二阶微分运算,检测出灰度值有较大变化的区域,即边缘。Laplace 算子对噪声比较敏感,因此通常会与高斯平滑滤波器结合使用。

Laplace 算子的应用可分为以下几个步骤:

1. 构建 Laplace 算子核

Laplace 算子有不同形式的离散卷积核,常用的有:4 邻域 Laplace 算子，8 邻域 Laplace 算子

2. 与原始图像进行卷积

将上述 Laplace 算子核与原始图像进行卷积运算,得到一个 Laplace 梯度图像。

3. 寻找过零点

在 Laplace 梯度图像中,边缘点附近会有一个由正值到负值或由负值到正值的过渡区,这个过渡区的中心就是过零点。通过检测梯度符号的变化,可以找到零交叉点,从而定位边缘。

D．Canny 边缘检测

Canny 边缘检测算法用于从图像中提取边缘信息[6]，被公认为边缘检测算法中较为优秀的一种。

Canny 算法的主要步骤如下:

1. 高斯滤波

为了减少图像噪声的影响,首先对图像进行高斯平滑滤波。高斯滤波器是一种线性平滑滤波器,通过计算像素及其邻域像素的加权平均值来消除图像噪声。

2. 计算梯度幅值和方向

分别计算图像每个像素位置的梯度幅值和方向。梯度幅值决定了像素点是否位于边缘上,梯度方向则用于确定边缘的方位。

2. 非极大值抑制

为了获得精确的边缘位置,对梯度幅值进行非极大值抑制。对于边缘上的每个像素,检查其梯度值是否大于沿梯度方向的两个相邻像素的梯度值。如果是,则保留该像素值;否则,将其置零。

4. 双阈值和连接边缘

使用高阈值和低阈值来确定真实的边缘。首先将高于高阈值的像素标记为强边缘像素,低于低阈值的像素标记为非边缘像素。对于介于两个阈值之间的像素,如果它与强边缘像素相连,则将其标记为边缘像素。

5. 最终得到边缘图像

最后,算法输出仅包含强边缘连接像素组成的二值边缘图像。

2.运行结果

使用 Canny 边缘检测 test3 和 test4 的结果如图 5 和图 9 所示，使用 laplace 边缘检测 test3 和 test4 的结果如图 6 和图 10 所示，使用 sobel 边缘检测 test3 和 test4 的结果如图 7 和图 11 所示，使用 unsharp masking 对 test3 和 test4 的结果如图 8 和图 12 所示。

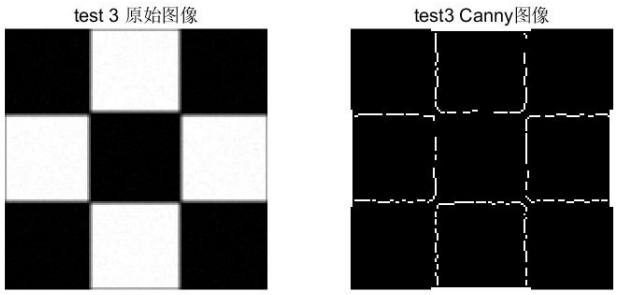

图 5. test3 的 Canny 边缘检测结果

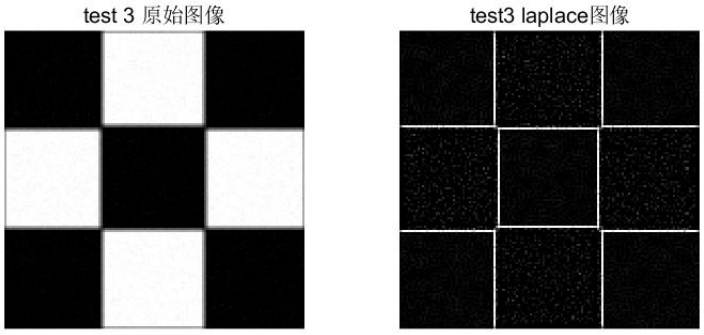

图 6. test3 的 laplace 边缘检测结果

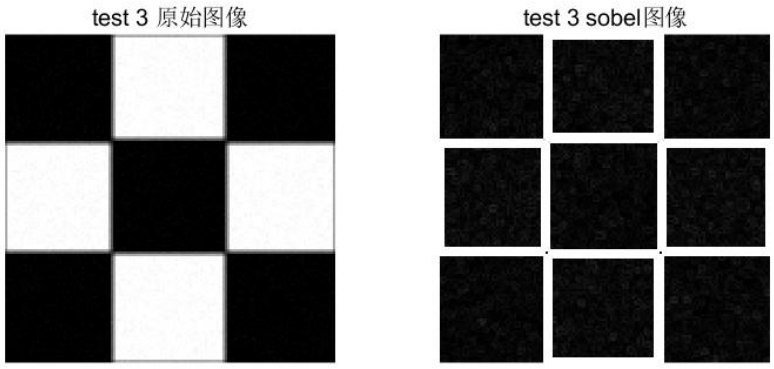

图 7. test3 的 sobel 边缘检测结果

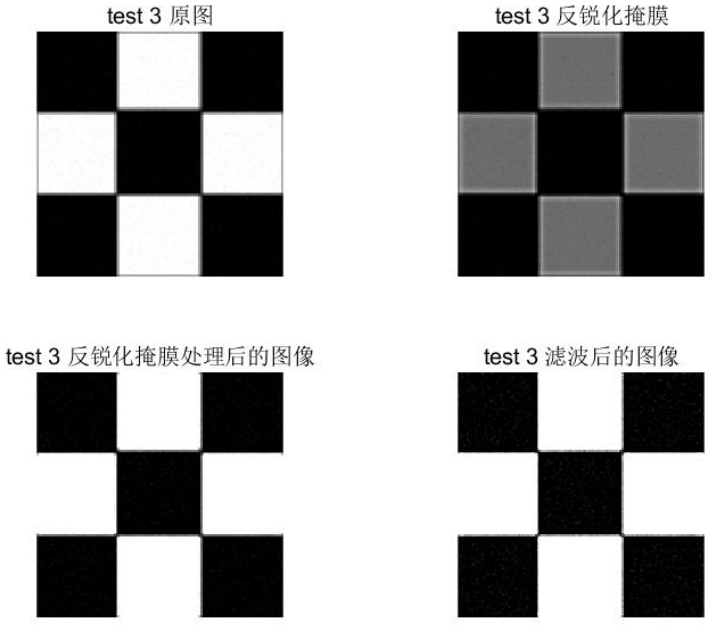

图 8. test3 的 unsharp masking 边缘检测结果

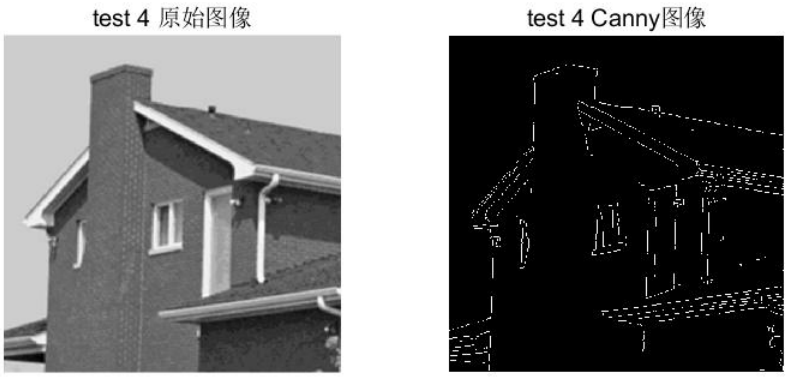

图 9. test4 的 Canny 边缘检测结果

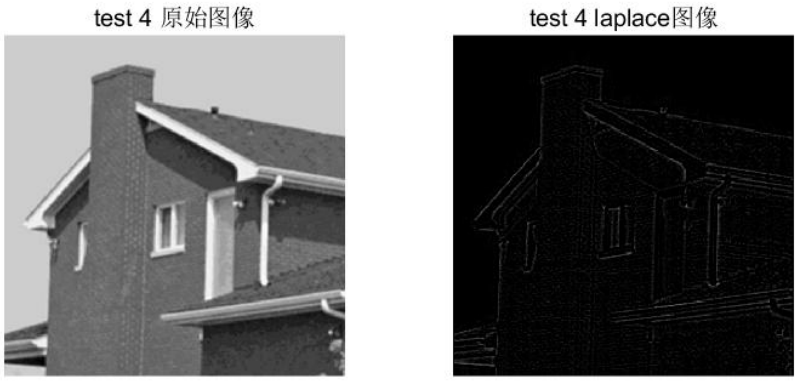

图 10. test4 的 laplace 边缘检测结果

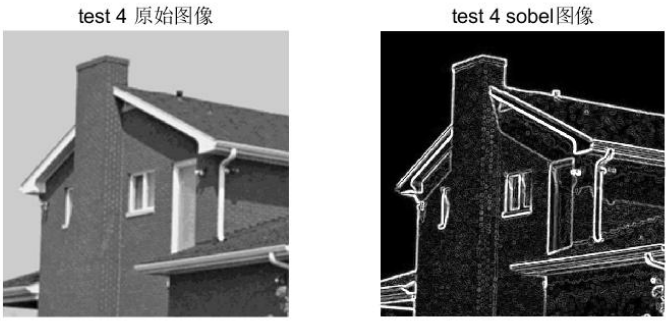

图 11. test4 的 sobel 边缘检测结果

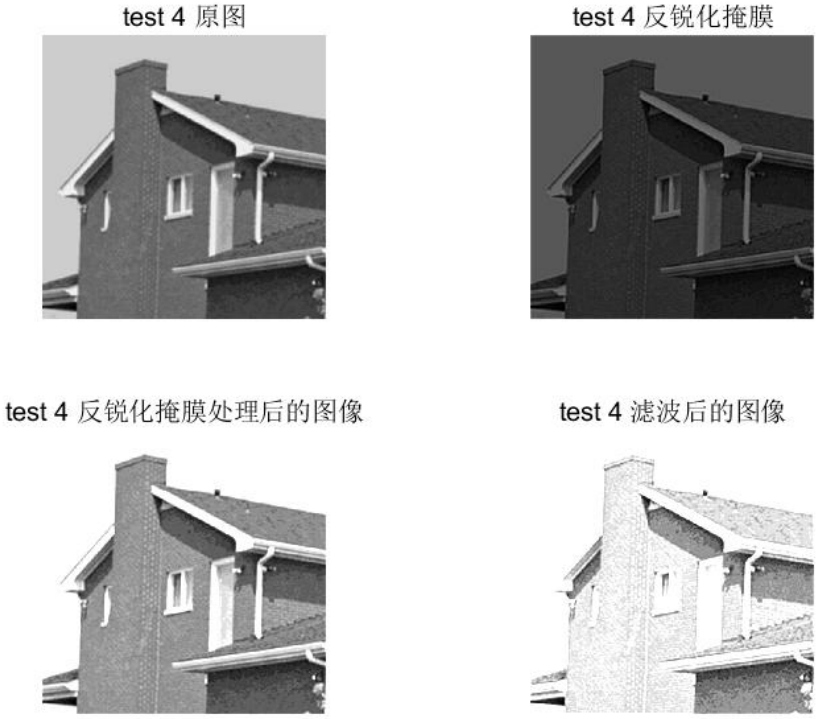

图 12. test4 的 unsharp masking 边缘检测结果

unsharp masking 的优点是简单、高效,能有效提高图像质量。缺点是可能引入噪声和幻影,需要合理选择参数。在实际应用中,可以结合不同的模糊核和参数配置,针对特定类型的图像获得最佳的锐化效果。

Sobel 算子相对简单高效,容易实现。内置了平滑噪声的能力,
能较好地检测出水平、垂直和对角线方向的边缘

Laplace 边缘检测算法对所有方向的边缘都有相同的响应，并且能够有效检测出亮边缘和暗边缘。但是，Laplace 对噪声非常敏感,需要先进行平滑滤波。

Canny 边缘检测算法具有良好的噪声抑制能力，对真实边缘的敏感度较高。缺点是计算量较大,对参数选择较为敏感。

由上图的变换效果可知，Canny 算法处理后的图像在一些连续的边缘轮廓处出现了断层现象（图 5，图 9）。造成这种断层现象的原因主要是：在双阈值抑制时,低于低阈值的像素会被抑制为非边缘点,高于高阈值的像素被保留为边缘点。对于中间阈值范围内的像素,只有与高阈值像素相连的部分才会被保留。这个过程可能会导致原本连续的边缘被分割开。通过调整阈值设置有望缓解这一问题。

对比图 6 和图 7,拉普拉斯算子检测到的边缘比 Sobel 算子更细。这主要是因为拉普拉斯算子使用的是二阶导数的近似,对灰度值的变化更加敏感,所以能够捕捉到一些比较细的边缘细节。而 Sobel 算子使用的是一阶导数近似,对较粗的边缘反应更为明显。

对比图 10 和图 11，拉普拉斯算子输出的对比度比 Sobel 算子输出的图像较暗。这种情况通常发生在图像的边缘过于明显、对比度过高的区域。拉普拉

斯算子对这种高对比度变化的敏感度非常高,输出值会趋于饱和,所以在这些区域会显得较暗。

而 Sobel 算子由于使用一阶导数近似,对这种高对比度的边缘响应相对平缓一些,所以在这些区域看起来对比度更高。说明在该类高对比区域,Sobel 算子对边缘的提取质量会更好。

参考文献

[1] “高斯滤波器,” 必应. Accessed: Mar. 13, 2024. [Online]. Available: https://cn.bing.com:9943/search?q=高斯滤波器&form=ANNTH1&refig=65f1a4fd676e4a5089ea44a0b3e2336a&pc=LCTS

[2] “图像处理学习笔记（十三）——平滑处理(中值滤波法)(理论篇),” 知乎专栏. Accessed: Mar. 13, 2024. [Online]. Available: https://zhuanlan.zhihu.com/p/343591732

[3] “Unsharp Mask(USM)锐化算法的的原理及其实现。_python 实现 photoshop 的 unsharp mask 功能基本原理-CSDN 博客.” Accessed: Mar. 13, 2024. [Online]. Available: https://blog.csdn.net/lz0499/article/details/80963696

[4] “OpenCV 中的边缘检测——Sobel 滤波器-CSDN 博客.” Accessed: Mar. 13, 2024. [Online]. Available: https://blog.csdn.net/qq_43010987/article/details/121641734

[5] “【数字图像处理】图像锐化：拉普拉斯算子（Laplacian）、高通滤波、Sobel 算子、Isotropic 算子、Prewitt 算子_用拉普拉斯算子对图像进行锐化-CSDN 博客.” Accessed: Mar. 13, 2024. [Online]. Available: https://blog.csdn.net/weixin_42415138/article/details/108574657

[6] “数字图像处理(20)：边缘检测算子(Canny 算子)_canny 算子是二阶导数的算子吗-CSDN 博客.” Accessed: Mar. 13, 2024. [Online]. Available: https://blog.csdn.net/zaishuiyifangxym/article/details/90142702