哈尔滨理工大学

实　验　报　告

课程名称：数字图像处理

预习报告	过程及记录	结果分析	实验总结	总评
教师签名：				

班　级 _____

学　号 _____

姓　名 _____

指导教师 范佳佳

教务处 印制

实验室安全管理个人注意事项

1.进入实验室工作、实验和研究人员必须进行实验室安全承诺，务必遵守学校及实验室各项规章制度和仪器设备操作规程。

2.熟悉紧急情况下的逃离路线和紧急应对措施，清楚急救箱、灭火器材、紧急洗眼装置和冲淋器的位置。

3.进行实验操作时，在做好个人防护的同时，要根据实验风险需要选择合适的实验个体防护用品。使用前应确认其使用范围、有效期及完好性等，熟悉其使用、维护和保养方法。

不得在实验室吸烟、饮食、储存食品、饮料等个人生活物品；不得做与实验、研究无关的事情。

4.触电事故特点：

4.1 被电击会导致人身伤害，甚至死亡；

4.2 短路有可能导致爆炸和火灾；

4.3 电弧或火花会点燃物品或者引燃具有爆炸性的物料；

4.4 冒失地开启或操作仪器设备可能导致仪器设备的损坏，使身体受伤；

4.5 电器过载会令其损坏、短路或燃烧。

5.触电事故的预防：

5.1 检查电线、插座和插头，一旦发现损坏要立即更换。

5.2 仪器设备开机前要先熟悉该仪器设备的操作规程，确认状态完好后方可接通电源。

5.3 当手、脚或身体沾湿或站在潮湿的地上时，切勿启动电源开关或接触电器用具。

6.触电事故应急措施：

6.1 使触电者脱离电源：应立即切断电源，可以采用关闭电源开关，用干燥木棍挑开电线或拉下电闸。救护人员注意穿上绝缘靴或站在干燥木板上，尽快使伤员脱离电源。

6.2 检查伤员：触电者脱离电源后，应迅速将其移到通风干燥的地方仰卧，并立即检查伤员情况。

6.3 急救并求医：根据受伤情况确定处理方法，对心跳、呼吸停止的，立即就地采用人工心肺复苏方法抢救，并及时拨打 120 急救电话。应坚持不懈地做心肺复苏，直到医生到达。

上述注意事项请仔细阅读后签字确认！

参加实验人员：_____（签名）

日　　期：　　年　　月　　日

实验名称	实验一 MATLAB 图像基础处理	时间	2024.10.26
		地点	新主楼 B207
同实验者		班组	

一、实验预习（准备）报告

1、实验目的

2、实验相关原理及内容

3、实验方法及步骤设计

4、实验设备仪器选择及材料

5、实验注意事项（包括实验过程、仪器设备、个人操作等方面）

6、实验思考题解答

二、实验过程及记录

本实验一共包括 4 个小实验，6 个程序代码。他们分别是①点对点的代数运算，它是运用图像代数运算，对两幅图像或多幅输入图像进行点对点的代数运算；②分段线性灰度级变换，它是改变分段函数参数大小来观察处理结果的；③非线性灰度级变换；④图像灰度直方图中，均衡化和规定化的对比。

①点对点的代数运算

该小实验是对图像进行代数运算，例如加减乘除。这里以加法为例，来对两幅图像进行不同的加法运算，通过视觉展示效果，便于理解图像加法运算的原理和应用。以下是代码实现：

实验一例 1 点对点的代数运算代码:

```
% 读取两幅图像
image1_add = imread('desert.jpg');  % 从文件中读取第一幅图像（背景图）
image2_add = imread('car.jpg');     % 从文件中读取第二幅图像（目标图）

% 截断求和
result1_add = imadd(image1_add, image2_add);  % 使用 imadd 函数进行截断求和，将两个图像的像素值逐点相加，超过 255 的部分被截断为 255

% 加权求和
alpha = 0.6;  % 设置加权系数 alpha，值在 0 到 1 之间，控制两幅图像的影响程度
result2_add = alpha * image1_add + (1 - alpha) * image2_add;  % 计算加权和，使用线性组合的方式调整图像亮度
```

```
% imlincomb 加权求和
result3_add = imlincomb(1 - alpha, image1_add, alpha, image2_add);  % 使用 imlincomb 函数进行加权求和，效果与上面相同

% 显示结果
subplot(321), imshow(image1_add), title("背景图");  % 在 321 子图中显示第一幅图像，并设置标题
subplot(322), imshow(image2_add), title("目标图");  % 在 322 子图中显示第二幅图像，并设置标题
subplot(323), imshow(result1_add), title("截断求和");  % 在 323 子图中显示截断求和结果，并设置标题
subplot(324), imshow(result2_add), title("加权求和");  % 在 324 子图中显示加权求和结果，并设置标题
subplot(325), imshow(result3_add), title("imlincomb 加权求和");  % 在 325 子图中显示 imlincomb 加权求和结果，并设置标题
```

在本次实验中，我对两幅图像进行了加法代数运算，主要包括截断求和和加权求和的实现。首先，使用`imread`函数读取了两幅图像，分别为背景图（`desert.jpg`）和目标图（`car.jpg`），这两幅图像将作为后续处理的输入。

接下来，进行截断求和的计算。通过调用`imadd`函数，我们对两幅图像进行了逐点相加。这一过程会自动处理像素值超出范围的情况，确保结果中的像素值不超过 255，从而避免图像失真。

此外，我还进行了加权求和，以便在图像融合中控制不同图像的影响程度。定义了一个权重系数`alpha`，设置为 0.6，这意味着背景图的权重为 0.6，目标图的权重为 0.4。通过线性组合公式，我们得到了加权和结果。这种方法使得在融合两幅图像时，能够根据需要调整它们的视觉效果。

此外，代码中还使用了`imlincomb`函数来实现加权求和，功能与上述线性组合相同，但在处理大型图像时通常会更高效。`imlincomb`提供了一种灵活的方式来处理多个输入图像的加权求和。

最后，使用`subplot`将背景图、目标图、截断求和结果、加权求和结果以及`imlincomb`加权求和结果展示在同一窗口中。每个子图都用`imshow`函数显示相应的图像，并通过`title`设置了明确的标题，便于观察和分析不同处理结果的差异。

②分段线性灰度级变换

该小实验是通过改变分段函数参数的大小，来实现不同的分段线性灰度级变换的效果，进而更好地理解灰度级变换的原理和应用。以下是代码实现：

实验一例 2 分段线性灰度级变换代码:

```
clear, clc, close all;  % 清除工作区、命令窗口和关闭所有图形窗口
Image = imread('couple.bmp');  % 读取输入图像文件 'couple.bmp'
[height, width] = size(Image);  % 获取图像的高度和宽度

% 定义第一个分段线性变换的参数
a1 = 80;
```

```
b1 = 200;
c1 = 30;
d1 = 220;
% 定义第二个分段线性变换的参数
a2 = 30;
b2 = 220;
c2 = 80;
d2 = 200;

% 初始化新图像矩阵
Newlmage1 = zeros(height, width);  % 创建一个与原图相同大小的零矩阵，用于存储变换结果 1
Newlmage2 = zeros(height, width);  % 创建另一个零矩阵，用于存储变换结果 2

% 对灰度值进行分段线性变换 for gray = 0:255  % 遍历所有可能的灰度值
    % 第一个分段线性变换
    if gray < a1
        newgray = c1 / a1 * gray;  % 在第一区间进行线性变换
    elseif gray < b1
        newgray = (d1 - c1) / (b1 - a1) * (gray - a1) + c1;  % 在第二区间进行线性插值
    else
        newgray = (255 - d1) / (255 - b1) * (gray - b1) + d1;  % 在第三区间进行线性插值
    end
    Newlmage1(Image == gray) = newgray / 255;  % 将计算出的新灰度值赋给新图像 1

    % 第二个分段线性变换
    if gray < a2
        newgray = c2 / a2 * gray;  % 在第一区间进行线性变换
    elseif gray < b2
        nemgray = (d2 - c2) / (b2 - a2) * (gray - a2) + c2;  % 在第二区间进行线性插值
    else
        newgray = (255 - d2) / (255 - b2) * (gray - b2) + d2;  % 在第三区间进行线性插值
    end
    Newlmage2(Image == gray) = newgray / 255;  % 将计算出的新灰度值赋给新图像 2
end

% 显示原图和变换后的图像
```

```
subplot(131), imshow(Image), title('原图-鄢梦雨，2112010523');  % 显示原图
subplot(132), imshow(Newlmage1), title('线性变换 1');  % 显示第一种线性变换结果
subplot(133), imshow(Newlmage2), title('线性变换 2');  % 显示第二种线性变换结果
```

在本次实验中，我通过改变分段函数的参数来观察不同灰度级变换的处理结果。首先，我们使用 imread 函数读取输入图像文件 couple.bmp，并获取图像的高度和宽度，以便为后续处理分配合适的矩阵大小。

接着，我定义了两个分段线性变换的参数集。对于每个变换，参数 a 和 b 确定了灰度值的区间，而参数 c 和 d 则定义了对应的输出灰度值。这些参数控制了输入灰度值到输出灰度值的映射关系，从而实现图像的不同视觉效果。

随后，我初始化了两个与原图大小相同的零矩阵 Newlmage1 和 Newlmage2，用以存储处理后的图像数据。在接下来的循环中，我遍历所有可能的灰度值（从 0 到 255），并根据每个灰度值的大小决定其所属的变换区间。对于每个灰度值，分别计算新的灰度值。

在第一个分段线性变换中，如果灰度值小于第一区间的阈值 a1，则使用线性变换公式直接计算新灰度值；如果在第一区间到第二区间之间，则通过线性插值计算新灰度值；最后，对于超过第二区间阈值的灰度值，则采用另一个线性插值公式进行计算。相同的逻辑也适用于第二个分段线性变换，只是使用了不同的参数。

在循环结束后，将每个灰度值对应的新值存储到新图像矩阵中。为了保证结果的正确显示，再将新灰度值归一化到[0, 1]区间。通过这种方式，能够深入理解分段线性灰度级变换的原理及其在图像处理中的应用效果。

③非线性灰度级变换

该小实验中我分别使用了对数变换、指数变换、幂变换下两个不同参数来观察非线性灰度级变换函数的效果，以下是代码实现：

实验一例 3 非线性灰度级变换代码:

```
clear, clc, close all;  % 清除工作区、命令窗口和关闭所有图形窗口
grayl = double(imread('couple.bmp'));  % 读取输入图像 'couple.bmp'，并将其转换为双精度浮点格式
c1 = 255 / log(256);  % 计算对数变换的常数因子
c2 = 255 / (exp(2.56) - 1);  % 计算指数变换的常数因子

% 对数变换
result1 = uint8(c1 * log(grayl + 1));  % 应用对数变换公式，并转换回 8 位无符号整数格式

% 指数变换
result2 = uint8((exp(grayl * 0.01) - 1) * c2);  % 应用指数变换公式，并转换为 8 位无符号整数格式

% 幂变换
gamma1 = 0.35;  % 第一个幂变换的伽马值
```

```
gamma2 = 2;     % 第二个幂变换的伽马值
result3 = grayl .^ gamma1;  % 应用第一个幂变换
result4 = grayl .^ gamma2;  % 应用第二个幂变换

% 归一化结果
result3 = result3 / max(result3(:));  % 将结果归一化到[0, 1]区间
result4 = result4 / max(result4(:));  % 将结果归一化到[0, 1]区间

% 显示处理后的图像
subplot(221), imshow(result1), title('对数变换');  % 显示对数变换的结果
subplot(222), imshow(result2), title('指数变换');  % 显示指数变换的结果
subplot(223)  imshow(result3), title('幂变换 r=0.35');  % 显示幂变换结果，伽马值为 0.35
subplot(224), imshow(result4), title('幂变换 r=2');  % 显示幂变换结果，伽马值为 2

% 保存结果图像
imwrite(result1, '116-5-1.jpg');  % 保存对数变换的结果图像
imwrite(result2, 'li6-5-2.jpg');   % 保存指数变换的结果图像
```

在本次实验中，我编写了一个非线性灰度级变换程序，通过对数变换、指数变换和幂变换来观察不同的非线性变换对图像的影响。

首先，使用`imread`函数读取图像文件`couple.bmp`，并将其转换为双精度浮点数，以便于后续的计算。在进行非线性变换之前，定义了用于对数和指数变换的常数因子`c1`和`c2`，它们的计算确保了变换后灰度值仍在[0, 255]的范围内。具体而言，`c1`是通过公式`255/log(256)`获得的，而`c2`则通过`255/(exp(2.56) - 1)`计算得出。

接下来，分别应用对数变换和指数变换。对数变换的公式是`result1 = uint8(c1 * log(grayl + 1))`，将输入灰度值经过对数函数转换后再乘以常数因子`c1`，最终结果转换为 8 位无符号整数格式。指数变换则通过`result2 = uint8((exp(grayl * 0.01) - 1) * c2)`实现，采用了指数函数处理输入灰度值，并同样乘以常数因子`c2`。

此外，还实现了幂变换。我们设置了两个伽马值，分别为 0.35 和 2，使用这些伽马值进行幂变换，为了确保这些变换的结果不会超出显示范围，对结果进行了归一化处理，使其值范围保持在[0, 1]。

④均衡化和规定化的区别

该小实验中我观察了两种处理方法在改善图像质量方面的不同效果。直方图均衡化主要用于增强图像的对比度，而直方图规定化则用于使图像的灰度分布符合特定目标，从而实现更好的视觉效果。以下是代码实现：

实验一例 4 均衡化和规定化的区别代码:

```
% 读取原始图像和目标图像
image = imread("Couple.bmp");  % 读取原始图像
image_gray = im2gray(image);    % 将彩色图像转换为灰度图像
```

```
hist = imhist(image_gray);        % 计算原始灰度图像的灰度直方图

target_image = imread("Couple.bmp");  % 读取用于直方图规定化的目标图像
target_gray = im2gray(target_image);   % 将目标图像转换为灰度图像

% ---- 直方图均衡化 ----
% 图像尺寸
[height, width] = size(image_gray);  % 获取灰度图像的高度和宽度
NewImage = zeros(height, width);      % 初始化均衡化后的图像矩阵
s = zeros(256, 1);                    % 初始化累积分布函数 (CDF) 数组

% 计算累积分布函数 (CDF)
s(1) = hist(1);                       % 第一个灰度级的 CDF 初始化
for i = 2:256
    s(i) = s(i - 1) + hist(i);        % 累加每个灰度级的频率
end
s = s / (height * width);             % 归一化 CDF
s = round(s * 255);                   % 将 CDF 映射到 0-255 的灰度级范围

% 应用均衡化映射到新图像
for i = 1:height
    for j = 1:width
        NewImage(i, j) = s(image_gray(i, j) + 1);  % 将原灰度值映射到均衡化后的灰度值
    end
end

NewImage = uint8(NewImage);  % 将均衡化结果转换为 8 位无符号整数格式

% ---- 直方图规定化 ----
specified_image = imhistmatch(image_gray, target_gray);  % 将原图像的灰度分布规定化到目标图像的灰度分布

% ---- 显示结果 ----
figure;

% 显示原始图像及其直方图
subplot(3, 2, 1), imshow(image_gray), title("原始图像");
% 显示目标图像及其直方图
subplot(3, 2, 2), imshow(target_gray), title("目标图像");

subplot(3, 2, 3), imshow(NewImage), title("均衡化后的图像");
subplot(3, 2, 4), imshow(specified_image), title("规定化后的图像");
```

```

subplot(3, 2, 5), imhist(NewImage), title("均衡化后图像的直方图");
subplot(3, 2, 6), imhist(specified_image), title("规定化后图像的直方图");
```

在本次实验中，我实现了图像的灰度直方图均衡化和规定化，以分析这两种图像处理技术对图像灰度分布的影响。

在直方图均衡化这里，初始化一个新图像矩阵，并计算累积分布函数（CDF）。CDF 的计算从直方图开始，逐步累加每个灰度级的频率，并将其归一化到[0, 255]范围。通过遍历原图像的每个像素，将原灰度值映射到均衡化后的灰度值，生成均衡化结果。在直方图规定化中，使用`imhistmatch`函数将原图像的灰度分布调整为目标图像的灰度分布，使得经过规定化处理的图像在灰度分布上与目标图像相似。

最后，通过`subplot 将原始图像、目标图像、均衡化后和规定化后的图像展示在一个窗口中，同时显示它们的直方图，便于直观比较。通过本次实验，深入理解了直方图均衡化和规定化的原理及其应用，观察了它们在改善图像质量方面的不同效果。

三、实验结果分析

①点对点的代数运算

我对两幅图像进行了加法代数运算，背景图为沙漠，目标图为小汽车，分别使用了截断求和、加权求和和 imlincomb 加权求和三种方法，得到了三张处理后的图片。下图 1 是实验所得的结果。

截断求和是将两幅图像的像素值逐点相加，并在像素值超过 255 时进行截断，确保结果仍在有效的灰度范围内。截断求和的结果通常会显示出图像的细节，但可能导致亮度的丢失，因为强烈的光源区域（如汽车）可能因超出范围而被压制。这种方法适合对比度较低的图像，但在细节保留方面可能存在一定局限性。

在加权求和中，我为背景图和目标图分配了不同的权重（例如，0.6 和 0.4），通过调整这两个权重，我们可以控制两幅图像在最终结果中的影响程度。这种方法通常能够更好地保留细节，因为可以根据需要突出某一图像的特征。在我的结果中，可以观察到小汽车的轮廓更加清晰，且与背景的融合更加自然。

使用 imlincomb 函数实现的加权求和方法与前一种方法相似，但它提供了更加灵活和简便的参数设置方式。这种方法同样能够控制图像融合的强度，结果可能与加权求和相似，但在处理细节和亮度分布方面可能会更加优化。最终图像展现了较好的细节和层次感，使得小汽车与沙漠背景的结合显得更加和谐。

通过对比三种加法代数运算的结果，可以发现截断求和适合简单的图像融合，但在细节保留上存在不足，加权求和和 imlincomb 加权求和更具灵活性，能够更好地控制图像的融合效果，适合需要突出特定元素的情况。在实际应用中，选择适当的加法运算方法取决于具体的图像内容和目标效果，尤其是在需要保留细节和层次感时，优先考虑加权求和的方法。

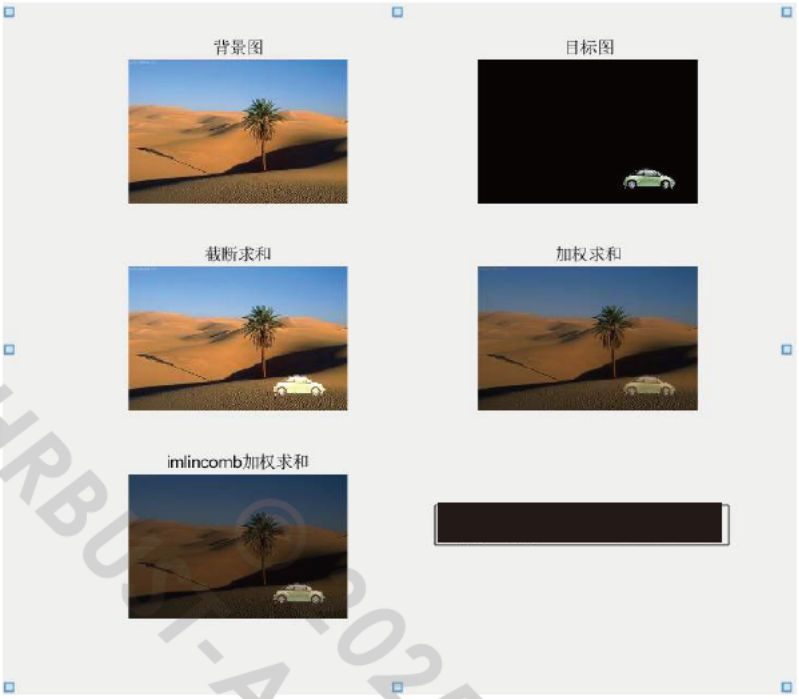

图 1 不同加法运算的效果

②分段线性灰度级变换

我对一张黑白照片进行了分段线性灰度级变换，照片中有一名男人和一名女人站在客厅里。通过两种不同的线性变换，得到了两种处理后的结果，如下图 2 所示。

第一种线性变换通过调整灰度级范围，使得背景变得更加黑暗。尽管这种处理可能增强了某些细节的对比度，但也导致了人物的轮廓变得不那么清晰。黑暗背景的增强掩盖了人物的部分细节，尤其是在较暗的区域，导致图像整体上看起来更为沉闷，缺乏层次感。这种处理方法可能适用于需要突出深色或阴影效果的场景，但在需要清晰展示主体的情况下，可能并不理想。

在第二种线性变换中，虽然图像的整体亮度有所提升，但伴随而来的是明显的噪点。这种噪点的产生可能是由于灰度级变换参数的设置不当，导致在高亮区域和暗部之间的过度增强。尽管人物的轮廓相对清晰，但噪点影响了图像的整体质量，使得视觉效果不够平滑且略显杂乱。这种变换在某些情况下可能有助于展示细节，但过多的噪点会干扰观众对图像内容的理解。

通过对比这两种分段线性灰度级变换的结果，可以发现第一种变换虽然增强了背景的对比度，但过度的黑暗处理削弱了人物的细节，导致视觉效果不佳。第二种变换在提升图像亮度的同时引入了噪点，使得整体画面显得不够干净。尽管人物轮廓更加明显，但噪点对图像的负面影响不容忽视。

图 2 不同分段线性变换的效果

③非线性灰度级变换

四种不同变换的结果如下图 3 所示，对数变换旨在增强低灰度值区域的细节，使得暗部细节更加明显。经过这种变换后，照片中的暗部区域（如阴影部分）得到了提升，使得人物的轮廓更加清晰。然而，由于对数变换对高灰度值的抑制，亮部可能会变得过于平坦，减少了对比度。

指数变换则相反，它强调高灰度值区域的细节，使得亮部更加突出。在这个变换下，照片中的亮部变得更加明亮，增加了视觉冲击力，但同时也可能导致暗部细节的损失。具体来说，人物的高光部分变得非常明显，但在阴影和较暗区域可能出现了信息的丢失。这种变换适合于需要突出高亮区域的场合，但在需要平衡暗部和亮部的情况下则不够理想。

幂变换通过调整幂指数，可以灵活控制图像的亮度和对比度。在使用较小的幂指数（如 0.35）时，暗部细节得到增强，整体图像变得更为明亮，且人物的轮廓更加清晰。这种参数适合需要提升暗部细节的场景。而在使用较大的幂指数（如 2）时，虽然整体亮度提升，亮部区域更加明亮，但可能导致图像的局部过曝，从而丢失细节。因此，幂变换的效果取决于幂指数的选择，适当的参数能够有效提升图像的视觉效果。

通过对比这些非线性灰度级变换的结果，我们可以得出以下结论：对数变换提升了暗部细节，但可能导致整体对比度不足；指数变换强调了亮部细节，但可能损失暗部信息，适合突出高亮区域。幂变换灵活性强，根据不同的幂指数可以实现不同的效果，但需谨慎选择，以避免过度曝光或信息丢失。

图 3 不同非线性变换的效果

④均衡化和规定化的区别

下图 4 是本小实验的结果图，均衡化旨在通过调整图像的灰度分布，使得图像的对比度得到提升。在处理后的图像中，暗部和亮部的细节均得到了增强，整体视觉效果更加清晰。人物的轮廓和背景的细节变得更加明显，使得图像的层次感增强。均衡化后的直方图呈现出更为均匀的分布，表明灰度级的利用更加充分，尤其是在原本较暗或较亮的区域得到了有效提升。

规定化则是将原图的灰度分布调整到与目标图像相似。在本实验中，规定化处理后的图像通常会展现出更符合目标图像的灰度特征。这种处理可以使得图像在某些特定条件下的可视性得到提升，例如，使得人物与背景的对比更加协调，视觉效果更加统一。规定化后的直方图显示出与目标图像相似的分布，这意味着它成功地调整了图像的灰度范围，从而增强了整体的视觉效果。

在两者的直方图中，均衡化后的直方图显示出均匀分布，表明灰度级利用更充分，有助于展示细节。规定化后的直方图则更接近目标图像的分布，体现了成功的灰度调整，但可能在整体对比度上不如均衡化明显。

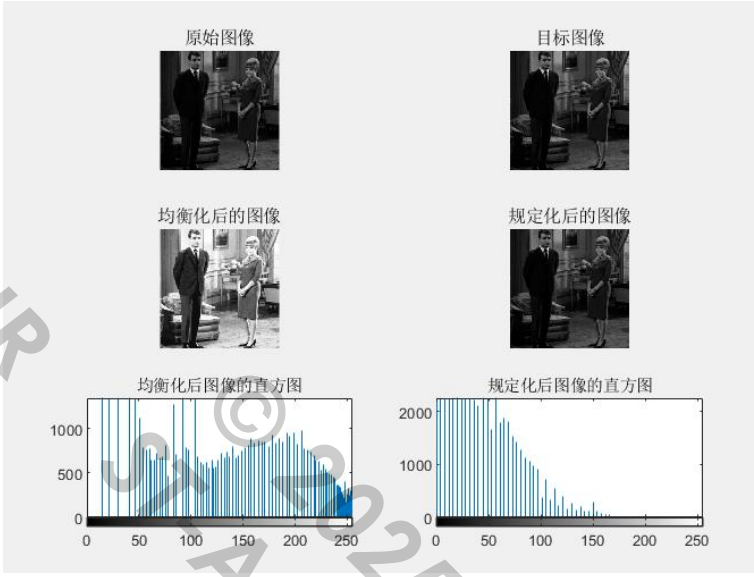

图 4 均衡化和规定化的对比

四、实验总结（200-300 字）

在实验一中，我通过不同的图像处理技术，观察了各自的效果与应用。首先，我运用图像代数运算对两幅图像进行了加法处理，得到了不同效果的合成图像，这一过程让我理解了图像叠加的原理及其在实际应用中的潜在价值。

接着，我通过改变分段函数的参数，观察到图像在明暗对比度和细节表现上的显著变化。这两组实验让我认识到，参数的选择在图像处理中的重要性，合理的参数设置能够有效提升图像质量。

此外，我进行了非线性灰度级变换，包括对数变换和幂变换，这让我体会到非线性方法在增强特定细节方面的优势。最后，通过对比灰度直方图均衡化和规定化的处理结果，我了解了均衡化在提升图像对比度上的效果及规定化在特定灰度分布匹配中的应用。

实验名称	实验二空域图像处理	时间	2024.10.26
		地点	新主楼 B207
同实验者		班组	

、实验预习（准备）报告

1、实验目的

2、实验相关原理及内容

3、实验方法及步骤设计

4、实验设备仪器选择及材料

5、实验注意事项（包括实验过程、仪器设备、个人操作等方面）

6、实验思考题解答

二、实验过程及记录

①均值滤波

该小实验是基于 3×3 的模板，来编写均值滤波的处理程序，处理含有加性高斯噪声和椒盐噪声的图像，观察处理结果。以下是代码实现：

实验二例 1 均值滤波代码:

```
clear, clc, close all; % 清空工作区，命令窗口，关闭所有图形窗口

% 读取原始图像并将其转换为双精度格式
Image = im2double(imread('girl.bmp'));

% 添加椒盐噪声，噪声比例为 1%
noiselsp = imnoise(Image, 'salt & pepper', 0.01);
% 添加高斯噪声
noiselg = imnoise(Image, 'gaussian');

% 对叠加椒盐噪声的图像应用 3x3 均值滤波
result1 = filter2(fspecial('average', 3), noiselsp);
% 对叠加椒盐噪声的图像应用 5x5 均值滤波
result2 = filter2(fspecial('average', 5), noiselsp);
% 对叠加高斯噪声的图像应用 3x3 均值滤波
result3 = filter2(fspecial('average', 3), noiselg);
% 对叠加高斯噪声的图像应用 5x5 均值滤波
result4 = filter2(fspecial('average', 5), noiselg);

% 创建图形窗口并显示图像
```

```
figure
subplot(131), imshow(Image), title('原图-鄢梦雨，2112010523'); % 显示原始图像
subplot(132), imshow(noiselsp), title('叠加椒盐噪声'); % 显示叠加椒盐噪声的图像
subplot(133), imshow(noiselg), title('叠加高斯噪声'); % 显示叠加高斯噪声的图像

% 创建另一个图形窗口并显示均值滤波结果
figure
subplot(221),imshow(result1),title('3x3 均值滤波抑制椒盐噪声');% 显示 3x3 均值滤波处理后的椒盐噪声图像
subplot(222),imshow(result2),title('5x5 均值滤波抑制椒盐噪声');% 显示 5x5 均值滤波处理后的椒盐噪声图像
subplot(223),imshow(result3),title('3x3 均值滤波抑制高斯噪声');% 显示 3x3 均值滤波处理后的高斯噪声图像
subplot(224),imshow(result4), itle('5x5 均值滤波抑制高斯噪声');% 显示 5x5 均值滤波处理后的高斯噪声图像
```

在本实验中，我们实现了对图像的均值滤波处理，以去除加性噪声。首先，使用 `imread` 函数读取原始图像，并将其转换为双精度格式以便后续处理。接着，我们通过 `imnoise` 函数向图像添加了两种类型的噪声：1%的椒盐噪声和高斯噪声。这使我们能够观察滤波器在不同噪声条件下的效果。随后，我们应用了均值滤波器。使用 `fspecial` 函数创建了 3×3 和 5×5 的均值滤波器，然后通过 `filter2` 函数分别对叠加噪声的图像进行处理。这一过程通过计算每个像素及其邻域的平均值来平滑图像，从而减少噪声影响。

最后，我们通过 `subplot` 和 `imshow` 函数在图形窗口中展示了处理结果。首先显示了原图及其添加噪声后的效果，然后依次展示了经过 3×3 和 5×5 均值滤波后的椒盐噪声和高斯噪声处理结果。通过这样的对比分析，可以清楚地观察到均值滤波器在抑制噪声方面的有效性，以及不同模板尺寸对图像处理结果的影响。

②中值滤波

该小实验是处理相同的含噪图像并与均值滤波进行比较;改变模板尺寸观察处理结果，以下是代码实现：

实验二例 2 中值滤波代码:

```
clear, clc, close all;  % 清除工作区、命令窗口和关闭所有图形窗口

% 读取图像并转换为双精度格式
Image = im2double(imread('girl.bmp'));

% 添加椒盐噪声到原图像，噪声比例为 1%
noiselsp = imnoise(Image, 'salt & pepper', 0.01);

% 添加高斯噪声到原图像
noiselg = imnoise(Image, 'gaussian');
```

```
11.
12. % 对叠加椒盐噪声的图像进行 3x3 中值滤波
13. result1 = medfilt2(noiselsp);  % 使用默认 3x3 模板
14.
15. % 对叠加椒盐噪声的图像进行 5x5 中值滤波
16. result2 = medfilt2(noiselsp, [5 5]);  % 指定 5x5 模板
17.
18. % 对叠加高斯噪声的图像进行 3x3 中值滤波
19. result3 = medfilt2(noiselg);  % 使用默认 3x3 模板
20.
21. % 对叠加高斯噪声的图像进行 5x5 中值滤波
22. result4 = m dfilt2(noiselg, [5 5]);  % 指定 5x5 模板
23.
24. % 显示原图像、椒盐噪声图像和高斯噪声图像
25. subplot(131), imsho (Image), title('原图-鄢梦雨，2112010523');
26. subplot(132), imshow(noiselsp)  title('叠加椒盐噪声');
27. subplot(133), imshow(noiselg), title('叠加高斯噪声');
28.
29. % 显示中值滤波处理后的结果
30. figure,
31. subplot(221), imshow(result1), title('3x3 中值滤波抑制椒盐噪声');
32. subplot(222), imshow(result2), title('5x5 中值滤波抑制椒盐噪声');
33. subplot(223), imshow(result3), title('3x3 中值滤波抑制高斯噪声');
34. subplot(224), imshow(result4), title('5x5 中值滤波抑制高斯噪声');
```

在本实验中，我们编写了一段 MATLAB 代码，使用中值滤波对含有椒盐噪声和高斯噪声的图像进行处理，以观察去噪效果。首先，我们读取了一幅名为“girl.bmp”的图像，并将其转换为双精度格式，以便于后续的处理。然后，我们向原图像添加了两种类型的噪声：1% 的椒盐噪声和高斯噪声，分别生成了噪声图像 noiselsp 和 noiselg。

在处理阶段，我们使用了中值滤波器来抑制噪声。对于叠加椒盐噪声的图像，我们首先应用了 3×3 的中值滤波器，并将结果存储在 result1 中；随后，使用 5×5 的中值滤波器处理相同的椒盐噪声图像，结果存储在 result2 中。对于叠加高斯噪声的图像，我们同样进行了 3×3 和 5×5 的中值滤波处理，结果分别存储在 result3 和 result4 中。

③使用一阶微分算子和二阶拉普拉斯算子进行图像锐化

该小实验是使用一阶微分算子和二阶拉普拉斯算子进行图像锐化观察处理结果，以下是代码实现：

实验二例 3 使用一阶微分算子和二阶拉普拉斯算子进行图像锐化代码:

```
1.  % 读取图像
2.  img = imread('frose.jpg');
3.  gray_img = rgb2gray(img); % 将图像转换为灰度图像
4.
5.  % 显示原始图像
6.  figure;
```

```
subplot(2, 2, 1);
imshow(img);
title('原始图像');

%% 一阶微分算子（Sobel 算子）
% 水平方向的 Sobel 算子
sobel_x = [-1 0 1; -2 0 2; -1 0 1];
% 垂直方向的 Sobel 算子
sobel_y = [-1 -2 -1; 0 0 0; 1 2 1];

% 应用 Sobel 算子
edge_x = imfilter(double(gray_img), sobel_x, 'replicate');
edge_y = imfilter(double(gray_img), sobel_y, 'replicate');

% 组合水平和垂直方向的梯度，得到边缘图
edge_img = sqrt(edge_x.^2 + edge_y.^2);

% 显示一阶微分结果
subplot(2, 2, 2);
imshow(uint8(edge_img));
title('一阶微分锐化结果 (Sobel) );

%% 二阶拉普拉斯算子
% 定义拉普拉斯算子
laplacian_filter = [0 -1 0; -1 4 -1; 0 -1 0];

% 应用拉普拉斯算子
laplacian_img = imfilter(double(gray_img), laplacian_filter,
'replicate');

% 图像锐化：将原图像加上拉普拉斯算子结果
sharpened_img = double(gray_img) - laplacian_img;

% 显示二阶拉普拉斯锐化结果
subplot(2, 2, 3);
imshow(uint8(sharpened_img));
title('二阶拉普拉斯锐化结果');

% 显示所有结果的组合图
subplot(2, 2, 4);
imshowpair(uint8(edge_img), uint8(sharpened_img), 'montage');
title('一阶和二阶锐化结果对比');
```

在本实验中使用水平和垂直方向的 Sobel 算子对图像进行滤波，分别得到水平方向和垂直

方向的边缘信息。通过计算梯度的幅值来合成边缘图，实现一阶微分锐化。使用拉普拉斯算子对图像进行滤波，提取二阶边缘信息，将原始图像减去拉普拉斯滤波的结果以实现图像的锐化。

④图像锐化和图像平滑的区别

该小实验是使用平滑滤波器，如均值滤波器、高斯滤波器，锐化滤波器，如 Sobel 算子、拉普拉斯算子，来观察图像锐化与图像平滑的区别，以下是代码实现：

实验二例 4 图像锐化和图像平滑的区别代码:

```
% 读取图像
img = imread('frose.jpg');
gray_img = rgb2gray(img); % 转换为灰度图像

% 显示原始图像
figure;
subplot(3, 3, 1);
imshow(gray_img);
title('原始图像');

%% 图像平滑处理

% 1. 使用 3x3 均值滤波器
mean_filter = fspecial('average', [3, 3]);
smoothed_mean = imfilter(gray_img, mean_filter, 'replicate');
subplot(3, 3, 2);
imshow(smoothed_mean);
title('均值平滑 (3x3)');

% 2. 使用 3x3 高斯滤波器
gaussian_filter = fspecial('gaussian', [3, 3], 0.5);
smoothed_gaussian = imfilter(gray_img, gaussian_filter,
'replicate');
subplot(3, 3, 3);
imshow(smoothed_gaussian);
title('高斯平滑 (3x3, σ=0.5)');

% 3. 使用 5x5 高斯滤波器（更强的平滑效果）
gaussian_filter_5x5 = fspecial('gaussian', [5, 5], 1);
smoothed_gaussian_5x5 = imfilter(gray_img, gaussian_filter_5x5,
'replicate');
subplot(3, 3, 4);
imshow(smoothed_gaussian_5x5);
title('高斯平滑 (5x5, σ=1)');

%% 图像锐化处理

```

```
% 4. 使用 Sobel 算子进行锐化（边缘检测）
sobel_x = fspecial('sobel'); % 水平方向 Sobel 算子
sobel_y = sobel_x'; % 垂直方向 Sobel 算子
edge_x = imfilter(double(gray_img), sobel_x, 'replicate');
edge_y = imfilter(double(gray_img), sobel_y, 'replicate');
edge_sobel = sqrt(edge_x.^2 + edge_y.^2);
subplot(3, 3, 5);
imshow(uint8(edge_sobel));
title('Sobel 锐化 (边缘检测)');

% 5. 使用 3x3 拉普拉斯算子进行锐化
laplacian_filter = [0 -1 0; -1 4 -1; 0 -1 0];
sharpened_laplacian = imfilter(double(gray_img), laplacian_filter,
'replicate');
sharpened_laplacian_img = double(gray_img) - sharpened_laplacian;
subplot(3, 3, 6);
imshow(uint8(sharpened_laplacian_img));
title('拉普拉斯锐化 (3x3) );

% 6. 使用增强型拉普拉斯算子进行更强的锐化
enhanced_laplacian_filter = fspecial('laplacian', 0.2); % 增强型拉
普拉斯
sharpened_enhanced_laplacian = imfilte (double(gray_img),
enhanced_laplacian_filter, 'replicate');
sharpened_enhanced_img = double(gray_img) -
sharpened_enhanced_laplacian;
subplot(3, 3, 7);
imshow(uint8(sharpened_enhanced_img));
title('增强拉普拉斯锐化');

%% 显示结果对比

% 显示原图和所有处理结果的并排对比图
subplot(3, 3, 8);
imshowpair(smoothed_gaussian, uint8(sharpened_laplacian_img),
'montage');
title('平滑 vs 锐化 (高斯 vs 拉普拉斯)');

subplot(3, 3, 9);
imshowpair(smoothed_gaussian_5x5, uint8(sharpened_enhanced_img),
'montage');
title('强平滑 vs 强锐化 (5x5 高斯 vs 增强拉普拉斯)');
```

三、实验结果分析

①均值滤波

下图 5 为对原图像分别叠加椒盐、高斯噪声的效果，可以发现椒盐噪声在图像中产生了明显的点状干扰，而高斯噪声则导致了较为均匀的亮度变化。处理这两种噪声的方式也有所不同，均值滤波对椒盐噪声通常更有效，因为它能够通过邻域平均来去除孤立的噪声点，而对于高斯噪声，可能需要结合其他滤波技术。

图 5 对原图像分别叠加椒盐、高斯噪声的效果

下图 6 为使用 3×3 和 5×5 均值滤波后的椒盐噪声和高斯噪声处理结果。使用 5×5 均值滤波器相比 3×3 均值滤波器，能够更有效地去除椒盐噪声。这是因为 5×5 滤波器考虑了更大范围的邻域像素，能更好地平均掉噪声点，减少孤立噪声对图像的影响。然而，过大的滤波器可能会导致图像细节的模糊。同样地，5×5 均值滤波器在去除高斯噪声时表现出更好的效果，能够更好地平滑图像，减少亮度变化带来的影响。但这也可能导致图像变得更模糊，细节部分可能会丢失。

图 6 使用 3×3 的模板进行均值处理

在处理椒盐噪声时，3×3 均值滤波器能够更好地保留图像的细节，因为它只对局部区域进行较小范围的平滑，而 5×5 滤波器可能导致一些细节丢失，尤其是在图像边缘和纹理处。对于高斯噪声，细节的保留影响较小，因为高斯噪声本身对整体图像结构的影响较为均匀。

使用 3×3 均值滤波器计算速度更快，因为其处理的像素数量较少。而 5×5 均值滤波器的计算量相对较大，可能导致处理时间增加。

②中值滤波

下图 7 是处理结果，可以看出中值滤波对高斯噪声的抑制效果较差，这是因为高斯噪声在图像中呈现的是较为连续的小波动，而非高对比度的随机噪声点。因此，即便使用 5x5 的模板，仍然会有一些噪声残留。中值滤波能在一定程度上平滑高斯噪声，但效果不如均值滤波，

因为均值滤波通过像素值的平均更适合处理高斯噪声。

而对椒盐噪声抑制，于 5x5 中值滤波器相比 3x3 具有更好的噪声抑制效果，因为更大的模板能够覆盖更多的噪声像素，处理后的图像更为平滑。

图 7 中值滤波处理效果

③使用一阶微分算子和二阶拉普拉斯算子进行图像锐化

下图 8 是处理结果，可以发现一阶微分算子（Sobel 算子）锐化效果结果图中，图像的边缘被突出显示，尤其是在物体轮廓和明显的亮度变化处。这种锐化方式适合用于提取边缘特征，但不太适合用于整体的图像锐化，因为它主要突出的是边缘区域，忽略了平滑区域的细节。

二阶拉普拉斯算子锐化效果结果图中，拉普拉斯算子增强了图像的细节，不仅是边缘区域，还有图像中其他细微的纹理区域。整体的锐化效果更加均匀，使得图像整体看起来更清晰。

一阶微分算子适合用于边缘增强，而二阶拉普拉斯算子适合于整体锐化。根据具体的应用需求，可以选择合适的算子，也可以将二者结合使用，进一步增强图像效果。

图 8 使用一阶微分算子和二阶拉普拉斯算子进行图像锐化的结果图

④图像锐化和图像平滑的区别

下图 9 是处理结果，可以发现平滑处理使图像变得柔和，细节模糊，适合去除噪声；而锐化处理则增强了图像的边缘和细节，使图像更加清晰，但可能会带来噪声的放大。

图 9 使用不同卷积模板运算的结果图

四、实验总结（200-300 字）

均值滤波一般不适合椒盐噪声，因为均值滤波会对模板内的值进行平均，导致边缘模糊且部分噪声残留。中值滤波对于椒盐噪声的去除效果更好，因为它保留了边缘并且能有效去除孤立的噪声点。对于高斯噪声，均值滤波通常比中值滤波效果更佳。均值滤波通过对像素值的平均处理，能够较好地平滑掉噪声。中值滤波在去除孤立噪声上较好，但整体平滑效果不如均值滤波。

无论是中值滤波还是均值滤波，增大模板尺寸通常会增强平滑效果。较大的模板能够更有效地抑制噪声，但会导致边缘细节的模糊。3x3 模板在噪声抑制和细节保留之间达到了平衡，而 5x5 模板则在平滑度上更强，对噪声较多的图像有更好的抑制效果，但模糊感也更明显。

哈尔滨理工大学

实　验　报　告

课程名称：数字图像处理

预习报告	过程及记录	结果分析	实验总结	总评
教师签名：				

班　级 [redacted]

学　号

姓　名

指导教师 范佳佳

教务处 印制

实验室安全管理个人注意事项

1.进入实验室工作、实验和研究人员必须进行实验室安全承诺，务必遵守学校及实验室各项规章制度和仪器设备操作规程。

2.熟悉紧急情况下的逃离路线和紧急应对措施，清楚急救箱、灭火器材、紧急洗眼装置和冲淋器的位置。

3.进行实验操作时，在做好个人防护的同时，要根据实验风险需要选择合适的实验个体防护用品。使用前应确认其使用范围、有效期及完好性等，熟悉其使用、维护和保养方法。

不得在实验室吸烟、饮食、储存食品、饮料等个人生活物品；不得做与实验、研究无关的事情。

4.触电事故特点：

4.1 被电击会导致人身伤害，甚至死亡；

4.2 短路有可能导致爆炸和火灾；

4.3 电弧或火花会点燃物品或者引燃具有爆炸性的物料；

4.4 冒失地开启或操作仪器设备可能导致仪器设备的损坏，使身体受伤；

4.5 电器过载会令其损坏、短路或燃烧。

5.触电事故的预防：

5.1 检查电线、插座和插头，一旦发现损坏要立即更换。

5.2 仪器设备开机前要先熟悉该仪器设备的操作规程，确认状态完好后方可接通电源。

5.3 当手、脚或身体沾湿或站在潮湿的地上时，切勿启动电源开关或接触电器用具。

6.触电事故应急措施：

6.1 使触电者脱离电源：应立即切断电源，可以采用关闭电源开关，用干燥木棍挑开电线或拉下电闸。救护人员注意穿上绝缘靴或站在干燥木板上，尽快使伤员脱离电源。

6.2 检查伤员：触电者脱离电源后，应迅速将其移到通风干燥的地方仰卧，并立即检查伤员情况。

6.3 急救并求医：根据受伤情况确定处理方法，对心跳、呼吸停止的，立即就地采用人工心肺复苏方法抢救，并及时拨打 120 急救电话。应坚持不懈地做心肺复苏，直到医生到达。

上述注意事项请仔细阅读后签字确认！

参加实验人员：_____（签名）

日　　期：　　年　　月　　日

实验名称	实验三频域图像处理	时间	2024.11.02
		地点	新主楼 B207
同实验者		班组	

、实验预习（准备）报告

1、实验目的

2、实验相关原理及内容

3、实验方法及步骤设计

4、实验设备仪器选择及材料

5、实验注意事项（包括实验过程、仪器设备、个人操作等方面）

6、实验思考题解答

二、实验过程及记录

①将图像旋转、缩放，观察频谱变化

该小实验是将图像旋转、缩放，观察频谱变化，结合傅立叶变换的性质进行分析结果，以下是代码实现：

实验三例 1 将图像旋转、缩放代码:

```
Image=rgb2gray(imread("block.bmp"));
[h,w,c]=size(Image);
scale=imresize(Image,0.5,"bilinear");
rotate=imrotate(Image,30,"bilinear","crop");
tform=affine2d([1 0 0;0 1 0;20 20 1]);
R=imref2d([h,w],[1 w],[1 h]);
trans=imwarp(Image,tform,"FillValues",0,"OutputView",R);
Originaldft=abs(fftshift(fft2(Image)));
Scaledft=abs(fftshift(fft2(scale)));
Rotatedft=abs(fftshift(fft2(rotate)));
Transdft=abs(fftshift(fft2(trans)));

subplot(221),imshow(Image),title("原图像");
subplot(222),imshow(scale),title("缩小变换图像");
subplot(223),imshow(rotate),title("旋转变换图像");
subplot(224),imshow(trans),title("平移变换");
```

②利用低通滤波器进行处理

该小实验是利用低通滤波器进行处理,以观察图像变化，以下是代码实现：

实验三例 2 利用低通滤波器进行处理代码:

```
Image=im2double(imread("frose.jpg"));
gray=rgb2gray(Image);
BW=edge(gray,"roberts");
```

```
H1=[1 0;0 -1]; H2=[0 -1;-1 0];
R1=imfilter(Image,H1);
R2=imfilter(Image,H2);
edgeImage=abs(R1)+abs(R2);
sharpImage=Image+edgeImage;

subplot(221),imshow(Image),title("原图像");
subplot(222),imshow(edgeImage),title("Roberts 梯度图像");
subplot(223),imshow(BW),title("Roberts 边缘检测");
subplot(224),imshow(sharpImage),title("Roberts 锐化图像");
```

③利用高通滤波器进行图像锐化

实验三例 3 利用高通滤波器进行处理代码:

```
Image=im2double(imread("frose.jpg"));
gray=rgb2gray(Image);
H=fspecial("laplacian",0);
R=imfilter(Image,H);
edgeImage=abs(R);
H1=[0 -1 0;-1 5 -1;0 -1 0];
sharpImage=imfilter(Image,H1);
BW=edge(gray,"zerocross",0.05,H);
subplot(221),imshow(Image),title("原图像");
subplot(222),imshow(edgeImage),title("Laplacian 滤波图像");
subplot(223),imshow(BW),title("Laplacian 边缘检测");
subplot(224),imshow(sharpImage),title("Laplacian 锐化图像");
imwrite(edgeImage,"li6-21-1.jpg");
imwrite(BW,"li6-21-2.jpg");
imwrite(sharpImage,"li6-21-3.jpg");
```

三、实验结果分析

①将图像旋转、缩放，观察频谱变化

下图 1 为处理结果可以发现当图像在空域中旋转一定角度时，其频谱图也会相应地旋转相同的角度。这是傅里叶变换的旋转不变性，即图像在空域的旋转会直接影响频域的角度变化。通过观察旋转后的频谱，可以验证这一性质。

当图像在空域中进行缩放时，其频谱会反向缩放。这是傅里叶变换的缩放性质：图像在空域放大，频谱在频域上会收缩，频率成分变得更加集中；而图像在空域缩小时，频谱会扩展，频率成分分布得更广。观察缩放后的频谱图，可以明显看到频率分布的变化。

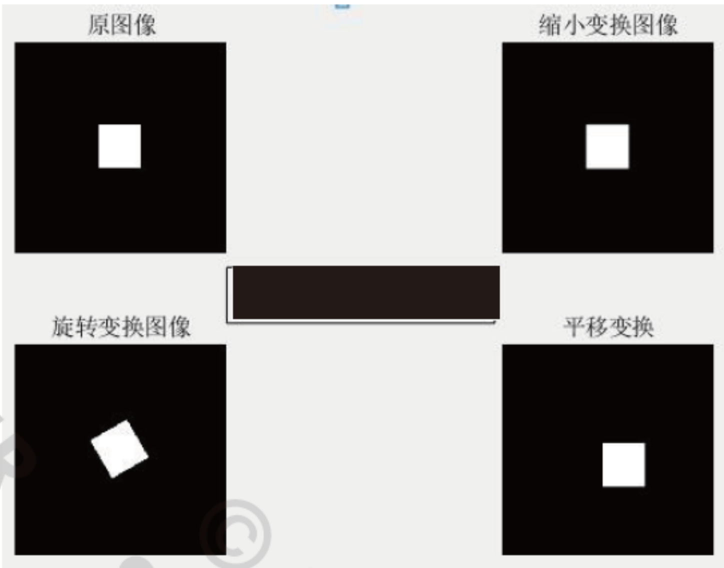

图 1 对原图像分别叠加椒盐、高斯噪声的效果

②利用低通滤波器进行处理

下图 2 为处理结果，通过计算图像的梯度，Roberts 算子能够捕捉到图像的边缘信息，生成的梯度图像反映了图像中强度变化的区域。二值化的边缘检测结果显示了图像的主要边缘特征。在原图像上叠加梯度图像后，边缘部分的细节被增强，图像显得更锐利，但同时也可能引入噪声。

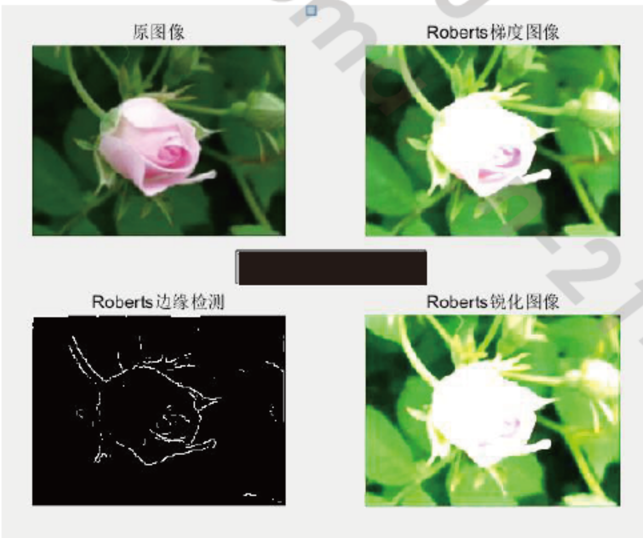

图 2 利用低通滤波器进行处理图

③利用高通滤波器进行图像锐化

下图 3 为处理结果，使用拉普拉斯算子生成的滤波图像主要突出了图像的边缘特征。因为拉普拉斯算子是二阶导数算子，对图像中快速变化的区域（如边缘）更敏感。通过零交叉法实现的边缘检测可以清晰地显示图像中的边缘结构，但会有少量噪声或不连续边缘。在原图上应用锐化滤波器后，图像的边缘和细节得到了增强，整体效果更加清晰。

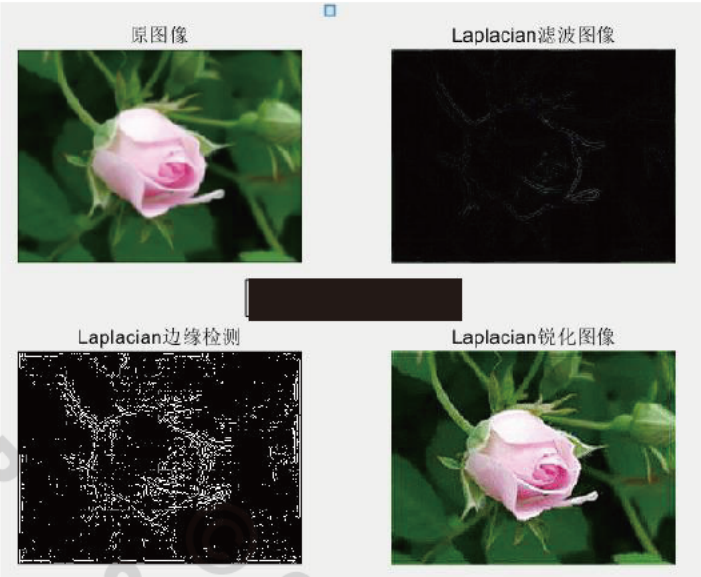

图 3 利用低通滤波器进行处理图

四、实验总结（200-300 字）

在以上三个实验中，我们利用不同的算子对图像进行了边缘检测与锐化处理，首先，通过 Roberts 算子实验，使用其简单的 2x2 滤波器来捕捉图像的边缘特征，实现了较为清晰的边缘检测效果。由于 Roberts 算子对噪声较敏感，适用于边缘明显、噪声较少的图像。接着，通过自定义卷积模板的实验，我们对比了不同模板的图像平滑与锐化效果，体会到平滑模板减少噪声的能力与锐化模板突出边缘的特性之间的差异。最后，拉普拉斯算子实验展示了二阶微分算子在边缘检测与锐化方面的应用，通过 Laplacian 滤波器对图像进行了零交叉边缘检测，进一步增强了边缘细节。

实验名称	实验四阈值分割与边缘检测	时间	2024.11.02
		地点	新主楼 B207
同实验者		班组	

一、实验预习（准备）报告

1、实验目的

2、实验相关原理及内容

3、实验方法及步骤设计

4、实验设备仪器选择及材料

5、实验注意事项（包括实验过程、仪器设备、个人操作等方面）

6、实验思考题解答

二、实验过程及记录

①最大类间方差法

实验四例 1 最大类间方差法代码:

```
I=imread('frose.jpg')
a=rgb2gray(I);
level = graythresh(a);
b=imbinarize(a,level);
subplot (131),imshow(I),title("原图像");
subplot (132), imshow (a),title("灰度图像");
subplot (133),imshow(b),title("outs 图像");
```

②利用一阶算子进行处理

实验四例 2 利用一阶算子进行处理代码:

```
X=imread('cameraman.jpg');
X1=rgb2gray(X);
BW_prewitt=edge(X1,'prewitt');%使用 prewitt 算子
BW_roberts=edge(X1,'roberts');%使用 roberts 算子
subplot(131),imshow(X),title('原图');
subplot(132),imshow(BW_prewitt),title('prewitt 算子');
subplot(133),imshow(BW_roberts),title('roberts 算子');
```

③利用二阶算子进行处理

实验四例 3 利用二阶算子进行处理代码:

```
X=imread('cameraman.jpg');
X1=rgb2gray(X);
w=[-2 -4 -4 -4 -2;-4 0 8 0 -4;-4 8 24 8 -4;-4 0 8 0 -4;-2 -4 -4 -4
-2];%LoG 算子的掩膜
BW_log=imfilter(X1,w);%使用 LoG 算子
subplot(121),imshow(X),title('原图');
```

```
6.   subplot(122),imshow(BW_log>200),title('log 算子');%显示二值图像，像素
     大于 200 则为 1
7.   subplot(133),imshow(BW_roberts),title('roberts 算子');
```

三、实验结果分析

①最大类间方差法

下图 4 为处理结果，可以发现 Otsu 法自动选择最佳阈值，将目标与背景区分开。处理后的图像中，目标区域更加突出，背景部分统一为较低的灰度值。但是 Otsu 法对噪声较为敏感，特别是在灰度分布不均匀时，可能导致分割误差。Otsu 法在单阈值情况下效果较好，但如果图像含有多个前景或背景灰度级较为复杂，单一阈值可能无法达到理想的分割效果。这时可以考虑局部阈值或其他多阈值方法进行比较。

图 4 最大类间方差法结果图

②利用一阶算子进行处理

下图 5 为处理结果，Roberts 算子更灵敏，适合细小边缘，但对噪声敏感。Prewitt 算子则提供平滑的边缘效果，更适合识别大区域边缘。Prewitt 算子抗噪能力较强，生成的边缘更加连续，而 Roberts 算子由于卷积核小，抗噪性稍差。Roberts 算子计算量更小，适合对计算速度要求较高的场景；Prewitt 算子计算量稍大，但更适合需要平滑、稳定边缘的图像。

图 5 利用一阶算子进行处理结果图

③利用二阶算子进行处理

下图 6 为处理结果，可以发现它对灰度值变化敏感，能有效突出强边缘，特别适合检测高频区域，使得图像中的边界清晰可见。拉普拉斯算子适合识别和突出较小的细节与纹理，因为它捕捉图像中变化剧烈的区域，这种效果在轮廓和纹理丰富的图像中尤为明显。然而，由于它无法分辨边缘的方向性，输出的边缘图像可能在一些区域产生噪声边缘，细节在增强的同时也可能会放大图像中的噪声。

图 6 利用二阶算子进行处理结果图

四、实验总结（200-300 字）

在本次实验中，分别使用了一阶算子的 Roberts 算子和 Prewitt 算子，以及二阶算子的拉普拉斯算子进行图像边缘检测与处理。各算子在边缘检测上表现出不同的特点和效果。

Roberts 算子采用小尺寸卷积核，适合检测图像的细小边缘，能够突出图像中的局部细节。Prewitt 算子相较于 Roberts 算子更能平滑处理边缘，生成连续性更好的边缘，适合噪声较少的场景。拉普拉斯算子作为二阶算子，在边缘锐化方面表现优异，但因其缺少方向性和对噪声的放大作用，通常需要搭配平滑滤波预处理。

总体而言，Roberts 算子和 Prewitt 算子在一阶导数算子中各有特点，适合不同需求的边缘检测；而拉普拉斯算子则更适合需要细致轮廓提取的场景。通过对比，能够更好地理解不同算子的应用场景和优劣。