成绩：

数字图像处理

实验报告

姓名：刘柏良

学号：2022302934

专业：软件工程

西北工业大学

2025 年 5 月 26 日

考核实验 目标识别

一、 实验目的

1. 对给定情景下的两张焊缝像进行图像分割，并对边缘进行提取；
2. 掌握图像处理方法的综合应用；
3. 边缘检测方法的应用。

二、 实验设备（环境）及要求

Matlab R2023，操作系统 win11

三、 实验原理及方法

经典边缘检测算子：

Canny 算子

原理：先用高斯滤波降噪，再计算梯度幅值和方向，接着非极大值抑制细化边缘，最后用滞后阈值确定边缘。

特点：边缘检测精度高、连续性好，对噪声有一定鲁棒性。

MATLAB 实现：edge(img_gray, 'canny')

Sobel 算子

原理：通过水平和垂直方向的梯度模板计算梯度，进而检测边缘。

特点：计算简单，对水平、垂直边缘敏感，对噪声有一定抑制。

MATLAB 实现：edge(img_gray, 'sobel')

LoG 算子

原理：先高斯平滑滤波降噪，再用拉普拉斯算子检测零交叉点作为边缘。

特点：能较好抑制噪声，检测零交叉点确定边缘，但计算复杂度相对高。

MATLAB 实现：edge(img_gray, 'log')

三种算子比较

精度：Canny > LoG > Sobel。

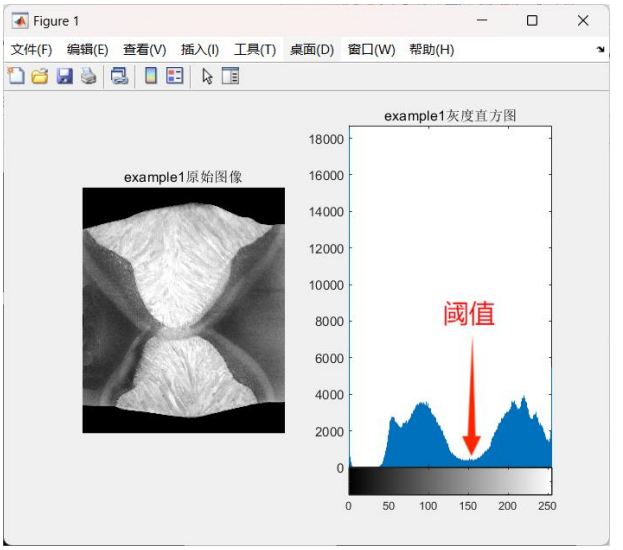
计算复杂度：Canny ≈ LoG > Sobel。

抗噪性：LoG > Canny > Sobel。

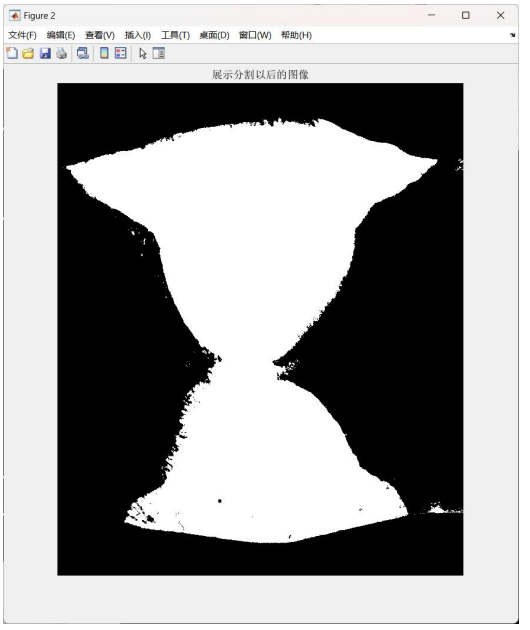
四、 设计思想

示例一：

1.观察 example1 的灰度直方图得知，其阈值在 150 附近，用窗口工具放大查看以及后期不断调参最后确定在 155



2.阈值分割法，把 example1 的图像灰度大于 155 的区域显示出来：

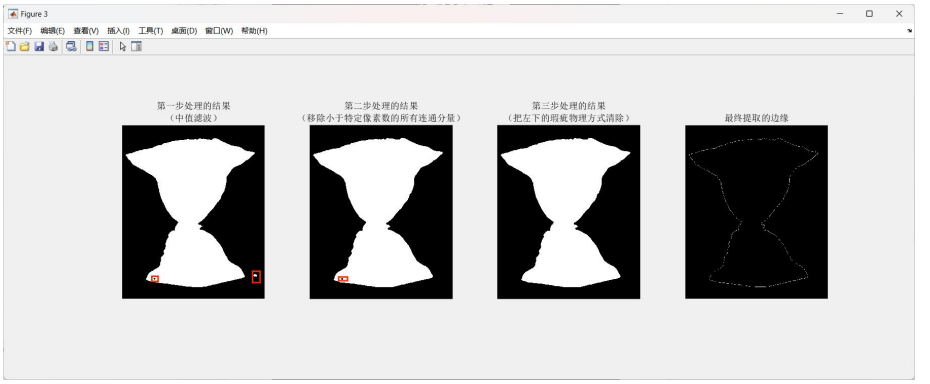


3.此时图像上还有很多噪音，边缘也不清晰尤其左下部分区域，需要进行平滑处理，我使用三种方法来处理：

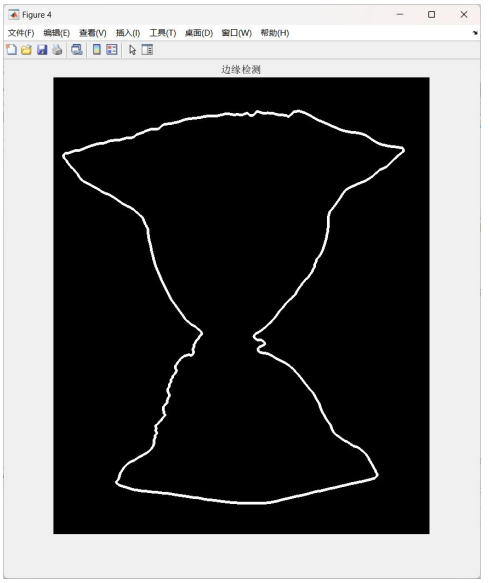
(1) 中值滤波处理，大窗口滤波器（12x12），处理噪声效果显著

(2) Bwareaopen 方法，可以把所有面积像素小于特定值的连通区域消除掉

(3) 人为补丁，在（2）处理以后仍有小片黑点，用窗口工具确定其坐标以后，人工涂白



边缘加粗加工以后的清晰边缘：



4.原图和边缘叠加展示

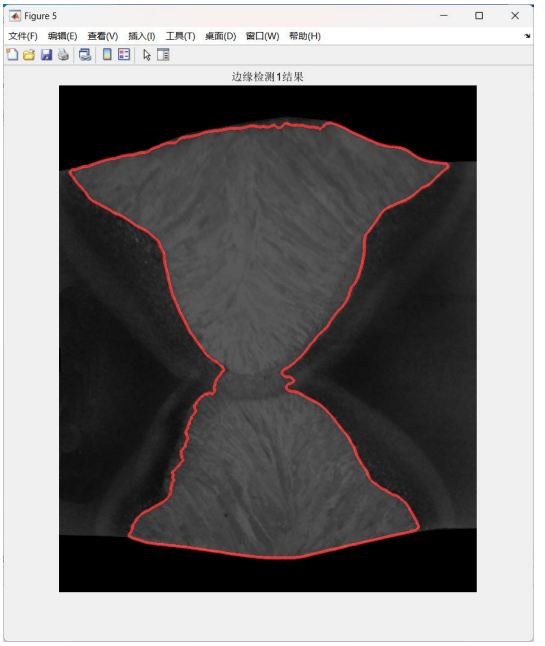
叠加方法是

image_final=image_1*alpha + image_2*(1-alpha)

alpha 是叠加权重。现知 example.jpg 是灰度图（只有黑白），而需要把边缘图像（也是灰度图）变成红色叠加上去，需要把两张图片都变为彩色图像格式才能用直接

相加的方法叠加。

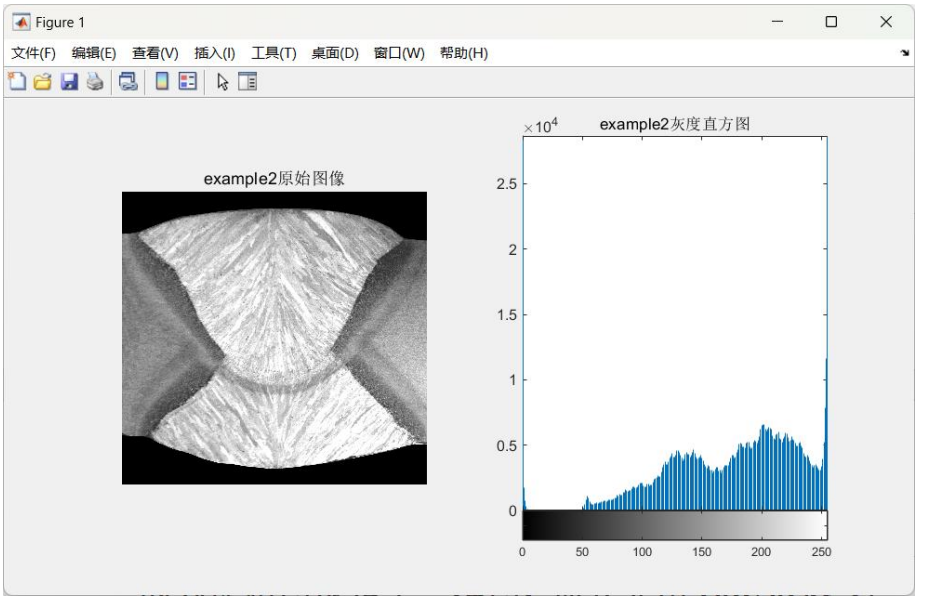
5.最终结果展示：



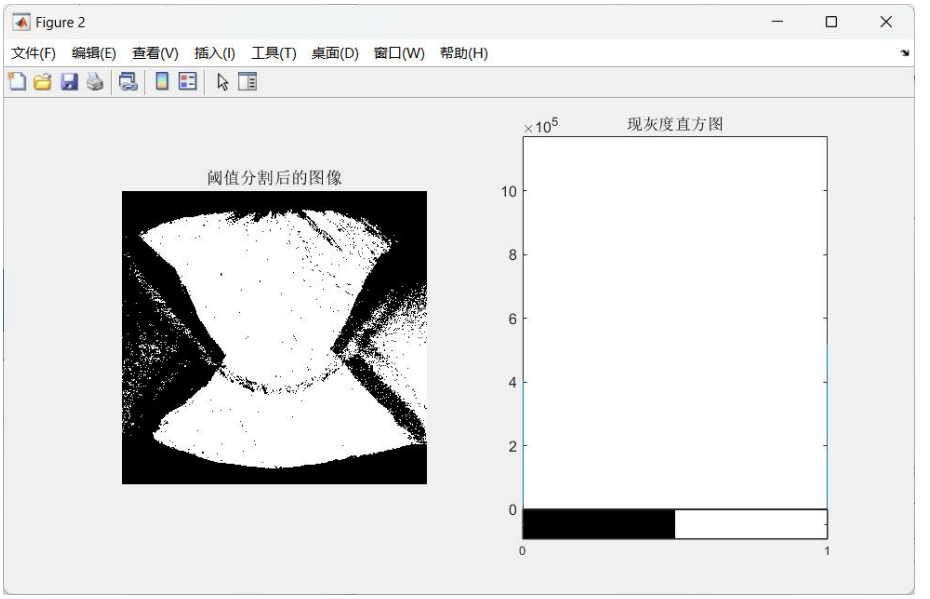
示例二：与示例一同理

1.先把原始图像进行阈值分割变成二值图：

因为原始图像是个三维图，需要先用 rgb2gray 转为灰度图；



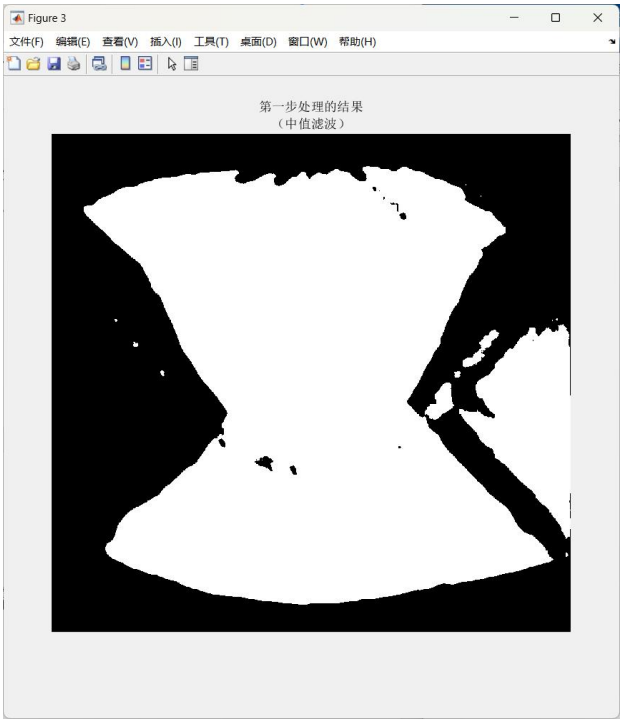
不易观察 example2.jpg 的阈值在哪里，只能不断调参试错将阈值定在 151 时有最佳表现，此时的分割后图像如下



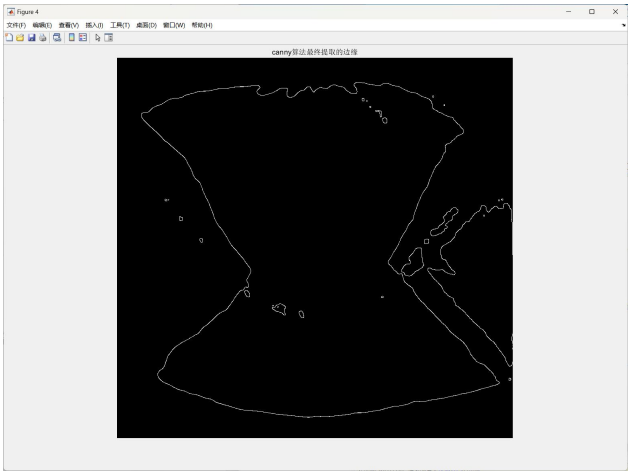
2.提取边缘操作

基本步骤：中值滤波处理、canny 算法检测边缘、删除断

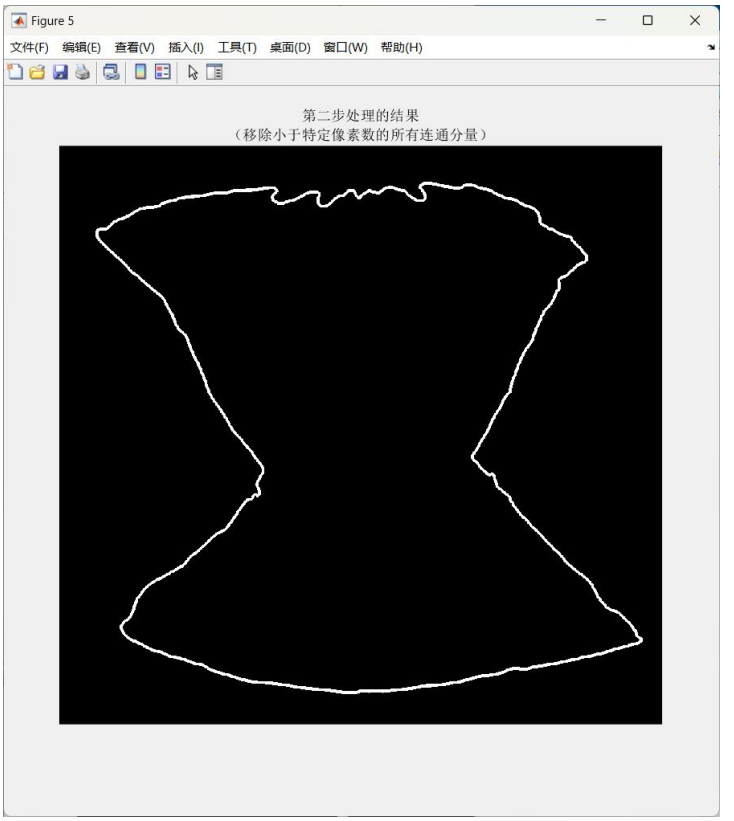
线和短线、加粗边缘方便展示。



此处经过多次二分法调参，加上思考和权衡以后，确定了使用边长为 16 的大窗口滤波器，大幅度削减噪声，代价是边缘也狭窄了。



Canny 算子整合了多项优化步骤，被广泛认为是最优的边缘检测算法，可以看到已经出现了目标边缘的完整轮廓，接下来要去掉图上无关的小线圈，仍旧用 bwareaopen 方法。最终的结果呈现如图：



3.将边缘检测图像与原图像叠加

和示例一方法一致，

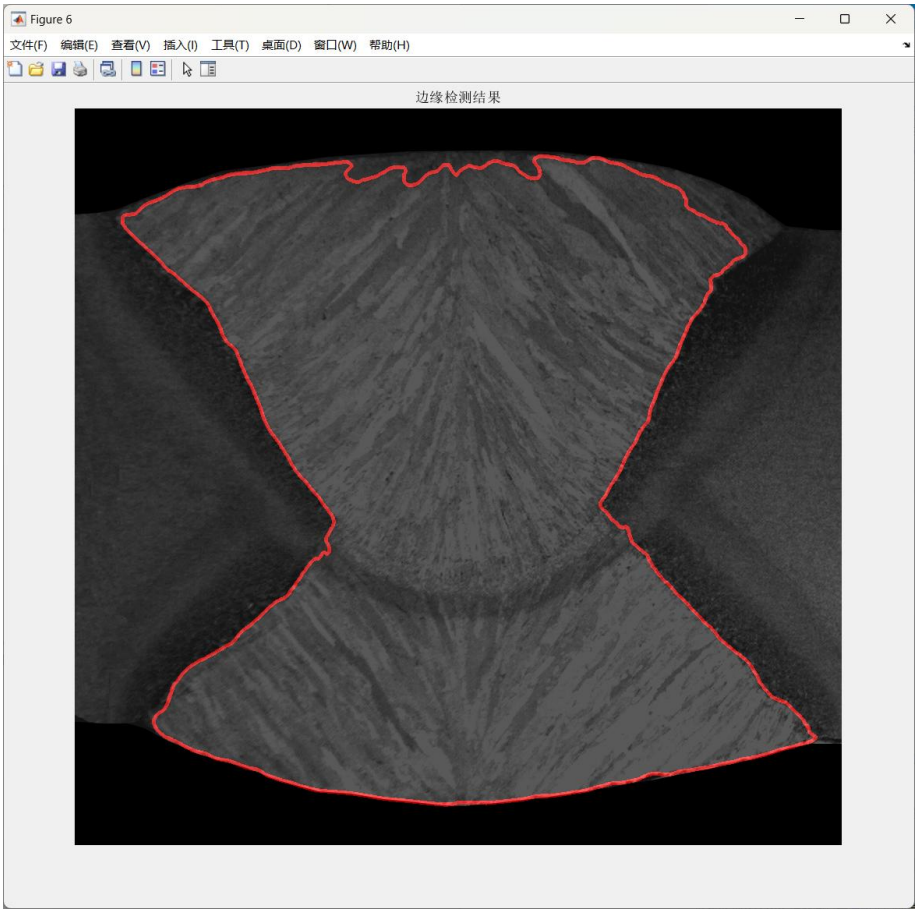
叠加方法是

image_final=image_1*alpha + image_2*(1-alpha)

alpha 是叠加权重。不同的是现知 example2.jpg 是彩色

图，只需要把边缘图像（也是灰度图）变成红色叠加上去，用直接相加的方法叠加即可。

4. 最终结果展示：



五、 程序代码及注释

示例一代码：

```
I_1=imread("example1.jpg"); %调入并显示原始图像 example1.jpg

%---------------图像分割（使用阈值分割技术）------------
figure(1);
```

```
subplot(1,2,1), imshow(I_1),title('example1 原始图像');
subplot(1,2,2), imhist(I_1);title('example1 灰度直方图');

J=I_1>155;%找到示例 1 阈值都在 155 附近，决定就取值 155
figure(2);
imshow(J),title('展示分割以后的图像');%展示分割以后的图像

%------------------提取边缘操作------------------
figure(3);
L_1 = medfilt2(J,[12 12]); %12×12 中值滤波处理噪声,去除图片细小噪声，更加纯净
subplot(141),imshow(L_1),title({'第一步处理的结果';'（中值滤波）'});
L_1=bwareaopen(L_1,500);%删除掉小区域的图像，进一步让图像纯净
subplot(142),imshow(L_1),title({'第二步处理的结果';'（移除小于特定像素数的所有连通分量）'});
L_1(734:743, 151:161) = 1;%为了确保边缘的纯净，直接把不能用 bwareaopen 删除的区域涂白
subplot(143),imshow(L_1),title({'第三步处理的结果';'（把左下的瑕疵物理方式清除）'});
BW1 = edge(L_1,'log');%log 算法检测边缘
subplot(144),imshow(BW1),title('最终提取的边缘');

% 加粗边缘检测结果
ee = strel('disk', 2); % 创建一个半径为 2 的圆形结构元素
BW1 = imdilate(BW1, ee); % 对边缘图像进行膨胀操作

%-----------------将边缘检测图像与原图像叠加-----------------
I_1 = repmat(I_1, [1, 1, 3]); % 如果是灰度图像，转换为彩色图像

% 创建一个全黑的彩色图像
I_edge = zeros(size(I_1), 'uint8');

% 将边缘部分设置为红色
I_edge(:, :, 1) = uint8(BW1) * 255; % 红色通道
I_edge(:, :, 2) = 0; % 绿色通道
I_edge(:, :, 3) = 0; % 蓝色通道
```

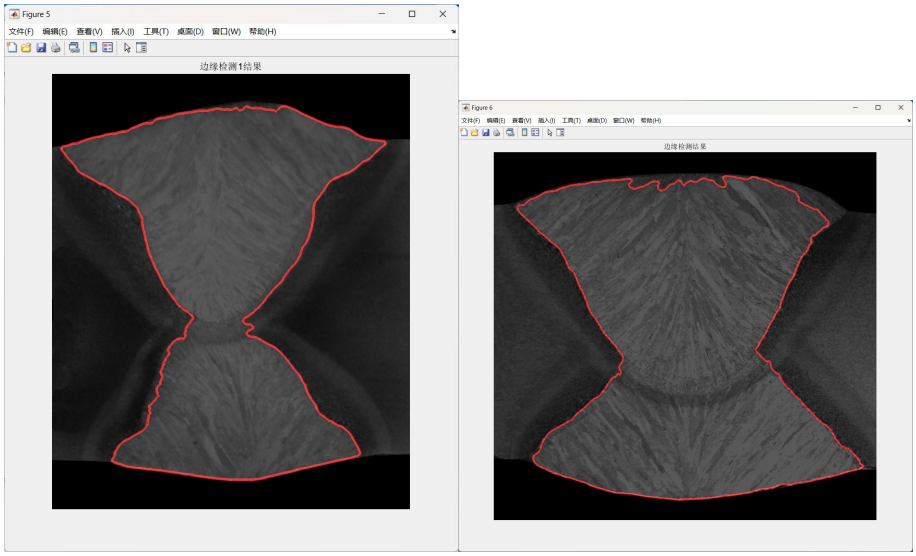
```
alpha = 0.65; % 叠加权重

I_overlay = I_1* (1 - alpha) + I_edge * alpha;
figure(4);
imshow(I_overlay), title('边缘检测 1 结果');
```

示例二代码：

```
I_2=imread("example2.jpg"); %调入并显示原始图像 example2.jpg
I_2_gray = rgb2gray(I_2);%I_2 不是二维图像，要转换

%-------------------二值图转化----------

figure,
subplot(121),imshow(I_2_gray),title('example2 原始图像');
subplot(122),imhist(I_2_gray);title('example2 灰度直方图');
I_2_gray=I_2_gray>151;%依旧使用阈值划分法，此处需要耐心调参
figure,
subplot(121),imshow(I_2_gray),title('阈值分割后的图像');
subplot(122),imhist(I_2_gray);title('现灰度直方图');

%------------------提取边缘操作------------------

M=16;%滤波器边长
L_1 = medfilt2(I_2_gray,[M M]); %中值滤波处理噪声,去除图片细小噪声，
更加纯净
figure,imshow(L_1),title({'第一步处理的结果';'（中值滤波）'});

BW1 = edge(L_1,'canny');%canny 算法检测边缘
figure,imshow(BW1),title('canny 算法最终提取的边缘');

L_1=bwareaopen(BW1,1400);%删除掉小区域的图像，进一步让图像纯净
% 加粗边缘检测结果
ee = strel('disk', 2); % 创建一个半径为 2 的圆形结构元素
L_1 = imdilate(L_1, ee); % 对边缘图像进行膨胀操作
figure,imshow(L_1),title({'第二步处理的结果';'（移除小于特定像素数
的所有连通分量）'});
```

```
%-----------------将边缘检测图像与原图像叠加-----------------
I_2_rgb = repmat(I_2_gray, [1, 1, 3]); % 如果是灰度图像，转换为彩色图像

% 创建一个全黑的彩色图像
I_edge = zeros(size(I_2), 'uint8');

% 将边缘部分设置为红色
I_edge(:, :, 1) = uint8(L_1) * 255; % 红色通道
I_edge(:, :, 2) = 0; % 绿色通道
I_edge(:, :, 3) = 0; % 蓝色通道

alpha = 0.65; % 叠加权重

I_overlay = I_2* (1 - alpha) + I_edge * alpha;
figure;
imshow(I_overlay), title('边缘检测 2 结果');
```

六、 实验结果



课堂实验一：MATLAB 数字图像基本处理

一、 实验目的

1．熟悉及掌握在 MATLAB 中能够处理哪些格式图像。
2．熟练掌握在 MATLAB 中如何读取图像。
3．掌握如何利用 MATLAB 来获取图像的大小、颜色、高度、宽度等等
相关信息。
4．掌握如何在 MATLAB 中按照指定要求存储一幅图像的方法。
5．图像间如何转化

二、 实验内容及步骤

1．利用 imread( )函数读取一幅图像，假设其名为 flower.tif，存入一个
数组中；
2．利用 whos 命令提取该读入图像 flower.tif 的基本信息；
3．利用 imshow()函数来显示这幅图像；
4．利用 imfinfo 函数来获取图像文件的压缩，颜色等等其他的详细信
息；
5．利用 imwrite()函数来压缩这幅图象，将其保存为一幅压缩了像素的
jpg 文件,设为 flower.jpg；语法：imwrite(原图像，新图像，‘quality’,q)，q 取
0-100。6．同样利用 imwrite()函数将最初读入的 tif 图象另存为一幅 bmp 图像，
设为 flower.bmp。
7．用 imread()读入图像：Lenna.jpg 和 camema.jpg；
8．用 imfinfo()获取图像 Lenna.jpg 和 camema.jpg 的大小；
9．用 figure,imshow()分别将 Lenna.jpg 和 camema.jpg 显示出来，观察
两幅图像的质量。
10．用 im2bw 将一幅灰度图像转化为二值图像，并且用 imshow 显示
出来观察图像的特征。

11．将每一步的函数执行语句拷贝下来，写入实验报告，并且将得到
第 3、9、10 步得到的图像效果拷贝下来。

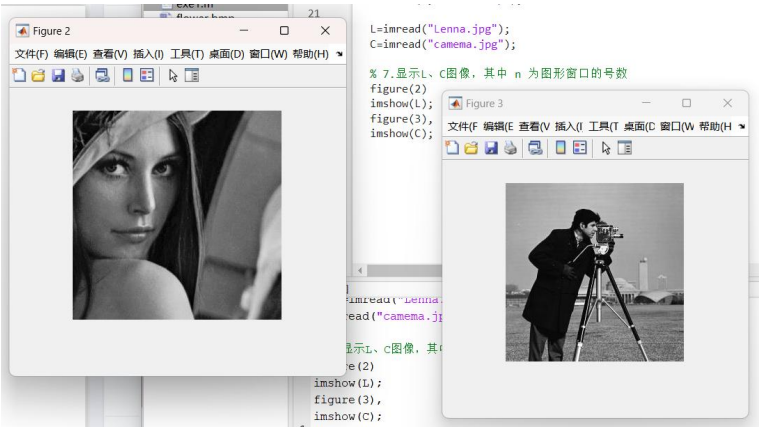
三、 实验代码及结果

代码：

```
% 1.利用 imread( )函数读取一幅图像，假设其名为 flower.tif，存入一个数组
I=imread('flower.tif');

%.2.利用 whos 命令提取该读入图像 flower.tif 的基本信息
whos I ;

%3.利用 imshow()函数来显示这幅图像；
imshow(I) ;
```

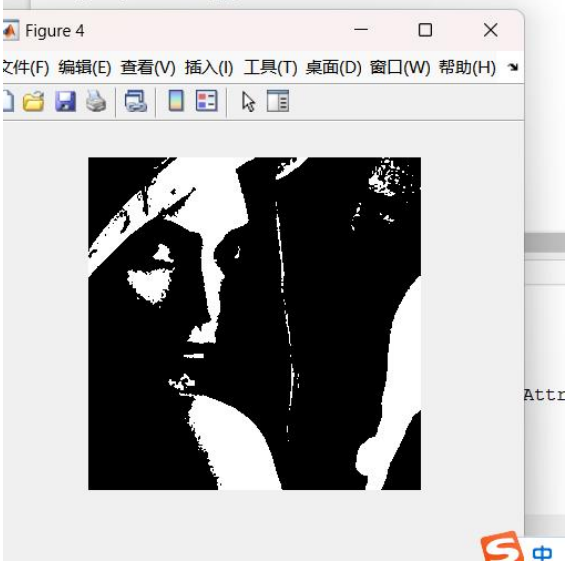

```
%4.利用 imfinfo 函数来获取图像文件的压缩，颜色等等其他的详细信息；
imfinfo flower.tif;

%5.利用 imwrite()函数来压缩这幅图象，
% 将其保存为一幅压缩了像素的 jpg 文件，设为 flower.jpg
imwrite(I,'flower.jpg','quality',75);
```

```
%6. 同样利用 imwrite()函数将最初读入的 tif 图象
% 另存为一幅 bmp 图像，设为 flower.bmp。
imwrite(I,'flower.bmp');

% 7 用 imread()读入图像：Lenna.jpg 和 camema.jpg：
L=imread("Lenna.jpg");
C=imread("camema.jpg");

%8 用 imfinfo()获取图像 Lenna.jpg 和 camema.jpg 的大小；
imfinfo Lenna.jpg;
imfinfo camema.jpg;

%9.用 figure,imshow()分别将 Lenna.jpg 和 camema.jpg 显示出来
% ，观察两幅图像的质量。
figure(2)
imshow(L);
figure(3),
imshow(C);
```




```
%10. 用 im2bw 将一幅灰度图像转化为二值图像，
% 并且用 imshow 显示出来观察图像的特征。
gg=im2bw(L); % Lenna.jpg 为例
figure, imshow(gg) % 显示二值图像
```

```
%10. 用 im2bw 将一幅灰度图像转化为二值图像，
% 并且用 imshow 显示出来观察图像的特征。
gg=im2bw(L); % 将图像转为二值图像
figure, imshow(gg) % 显示二值图像
```



四、 思考题

(1) 简述 MatLab 软件的特点。

强大的数值计算能力：支持矩阵运算、数值积分、微分方程求解等。

高级的图形可视化：可以生成二维和三维图形，支持多种图形类型。

易于使用的编程语言：语法简洁，内置大量函数和工具箱，支持多种应用领域。

良好的可扩展性和兼容性：支持与其他语言（如 C、Python）接口，兼容多种数据格式。

交互式环境：提供交互式命令窗口，方便实验和调试。

(2) MatLab 软件可以支持哪些图像文件格式?

bmp、tif、jpg、png、gif、pcx、mat 等

(3) 说明函数 imread 的用途格式以及各种格式所得到图像的性质。

用途：读取图像文件，返回图像数据矩阵。

格式：

A = imread(filename)：返回图像数据矩阵 A。

[A, map] = imread(filename)：对于索引图像，返回图像数据矩阵 A 和颜色映射表 map。
图像性质：
灰度图像：二维矩阵，值范围 0 到 255（8 位）或 0 到 65535（16 位）。
彩色图像：三维矩阵，第三维表示 RGB 通道，值范围 0 到 255（8 位）。
索引图像：二维矩阵，值为索引，对应颜色映射表 map。

(4) 为什么用 I = imread(‘lena.bmp’) 命令得到的图像 I 不可以进行算术运算
原因：I 是 uint8 类型，值范围 0 到 255，算术运算结果会被截断
解决方法：使用 im2double 将 I 转换为 double 类型，进行精确的算术运算。

课堂实验二：图像增强—空域滤波

一、 实验目的

进一步了解 MatLab 软件/语言，学会使用 MatLab 对图像作滤波处理，
使学生有机会掌握滤波算法，体会滤波效果。
了解几种不同滤波方式的使用和使用的场合，培养处理实际图像的能
力，并为课堂教学提供配套的实践机会。

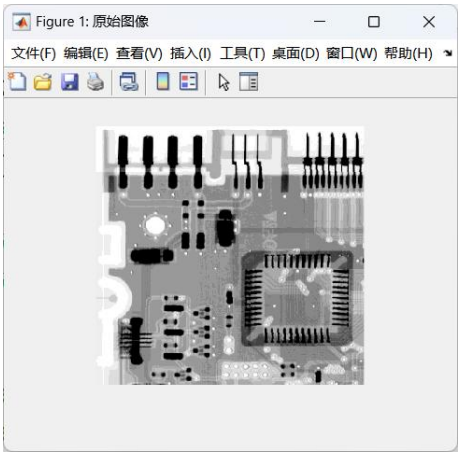
二、 实验内容及步骤

a) 调入并显示原始图像 Sample2-1.jpg 。
b) 利用 imnoise 命令在图像 Sample2-1.jpg 上加入高斯(gaussian) 噪声
c)利用预定义函数 fspecial 命令产生平均(average)滤波器
d）分别采用 3x3 和 5x5 的模板，分别用平均滤波器以及中值滤波器，
对加入噪声的图像进行处理并观察不同噪声水平下，上述滤波器处理的结
果;
e）选择不同大小的模板，对加入某一固定噪声水平噪声的图像进行处
理，观察上述滤波器处理的结果。
f）利用 imnoise 命令在图像 Sample2-1.jpg 上加入椒盐噪声(salt &
pepper)
g）重复 c)~ e）的步骤
h）输出全部结果并进行讨论。

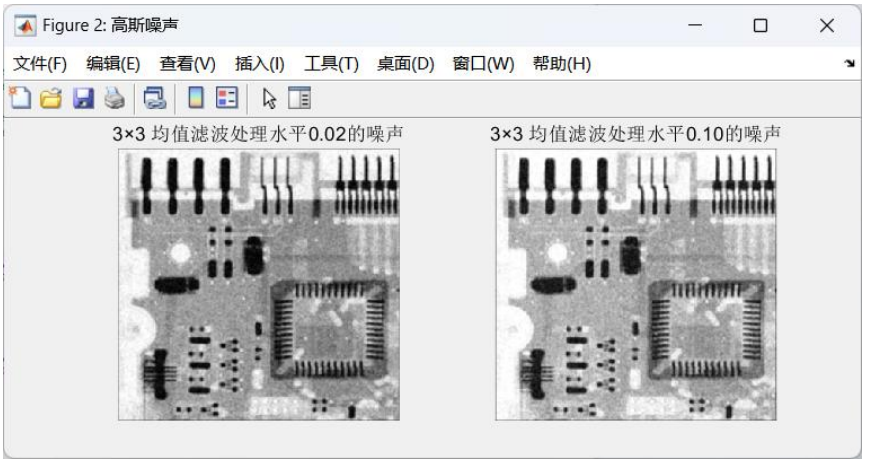
三、 实验代码及结果

分别讨论并展示结果，实验中把噪声水平分别设为 0.02 和 0.10

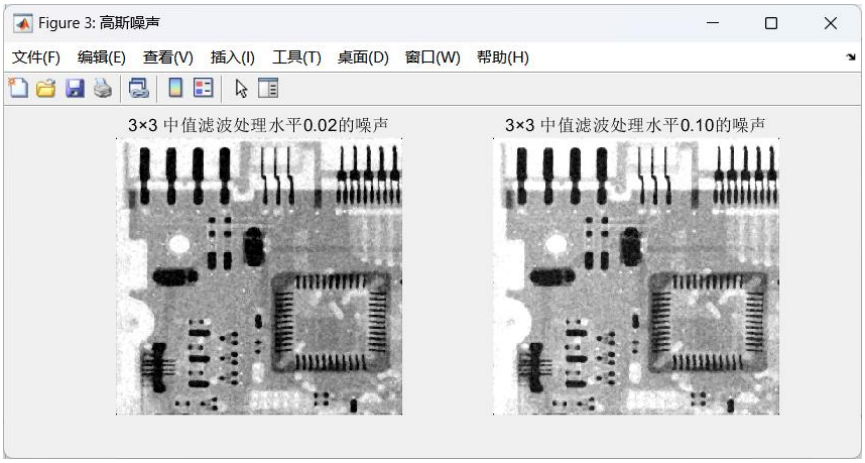
```
I=imread("Sample2-1.jpg"); %调入并显示原始图像 Sample2-1.jpg
figure('Name','原始图像')
imshow(I);
```




```
J_1 = imnoise(I,'gaussian',0.02);%利用 imnoise 命令在图像
Sample2-1.jpg 上加入高斯(gaussian) 噪声
J_2 = imnoise(I,"gaussian",0.1);%加入更高水平的高斯噪声

ave_1=fspecial('average',3); %产生 3x3 的平均滤波器
ave_2=fspecial('average',5); %产生 5x5 的平均滤波器

%3x3 模版下:
K_1 = filter2(ave_1,J_1)/255; %3×3 均值滤波处理水平 0.02 的噪声
K_2 = filter2(ave_1,J_2)/255; %3×3 均值滤波处理水平 0.10 的噪声

figure('Name','高斯噪声')
subplot(1,2,1); imshow(K_1);title('3×3 均值滤波处理水平 0.02 的噪
声')
subplot(1,2,2); imshow(K_2);title('3×3 均值滤波处理水平 0.10 的噪
声')
```

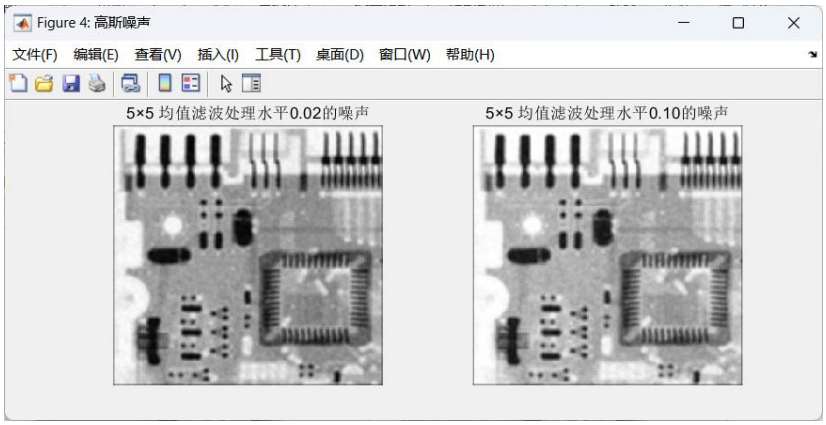


```
L_1 = medfilt2(J_1,[3 3]); %3×3 中值滤波处理水平 0.02 的噪声
L_2 = medfilt2(J_2,[3 3]); %3×3 中值滤波处理水平 0.10 的噪声

figure('Name','高斯噪声')
subplot(1,2,1); imshow(L_1);title('3×3 中值滤波处理水平 0.02 的噪声')
subplot(1,2,2); imshow(L_2);title('3×3 中值滤波处理水平 0.10 的噪声')
```

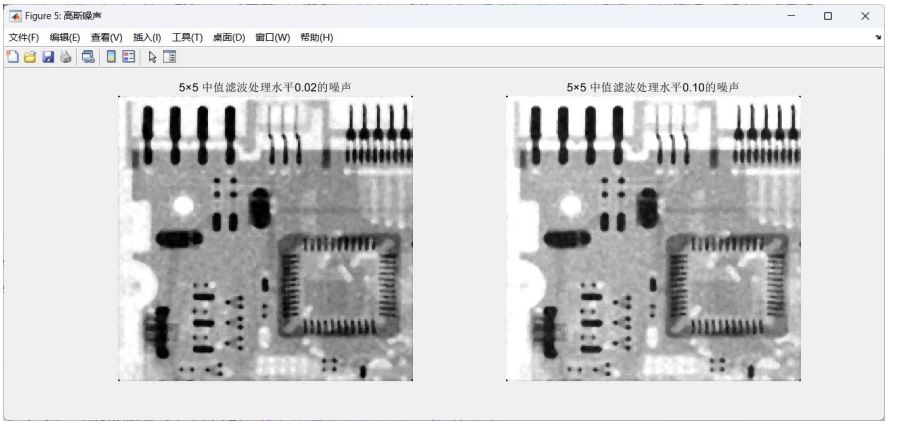


%5x5 模版下：

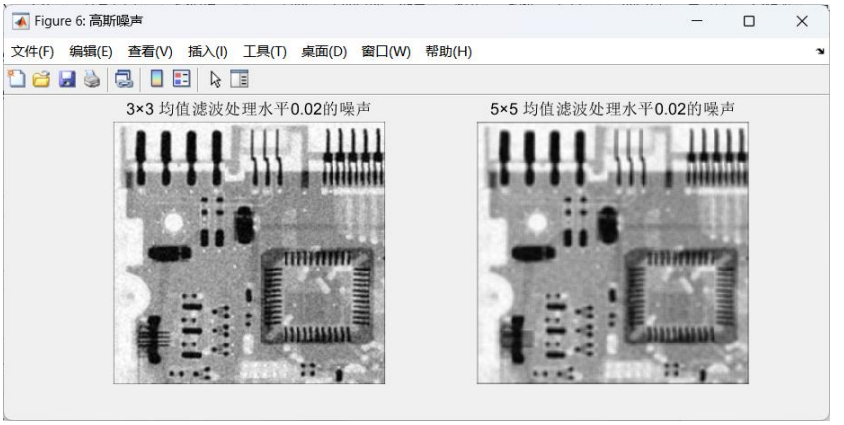
```
M_1 = filter2(ave_2,J_1)/255; %5×5 均值滤波处理水平 0.02 的噪声
M_2 = filter2(ave_2,J_2)/255; %5×5 均值滤波处理水平 0.10 的噪声

figure('Name','高斯噪声')
subplot(1,2,1); imshow(M_1);title('5×5 均值滤波处理水平 0.02 的噪声')
subplot(1,2,2); imshow(M_2);title('5×5 均值滤波处理水平 0.10 的噪声')
```



```
N_1 = medfilt2(J_1,[5 5]); %5×5 中值滤波处理水平 0.02 的噪声
N_2 = medfilt2(J_2,[5 5]); %5×5 中值滤波处理水平 0.10 的噪声

figure('Name','高斯噪声')
subplot(1,2,1); imshow(N_1);title('5×5 中值滤波处理水平 0.02 的噪声')
subplot(1,2,2); imshow(N_2);title('5×5 中值滤波处理水平 0.10 的噪声')
```

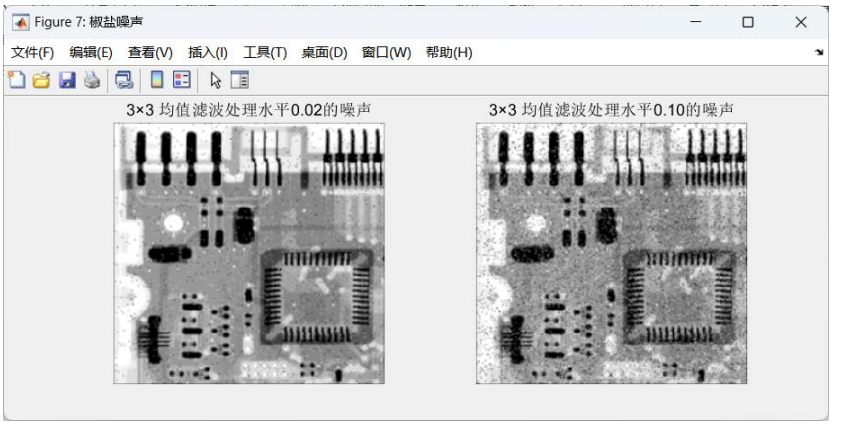



```
%对比 3x3 和 5x5 模版下同一噪声水平的处理
figure('Name','高斯噪声')
subplot(1,2,1); imshow(K_1);title('3×3 均值滤波处理水平 0.02 的噪声')
subplot(1,2,2); imshow(M_1);title('5×5 均值滤波处理水平 0.02 的噪声')
```

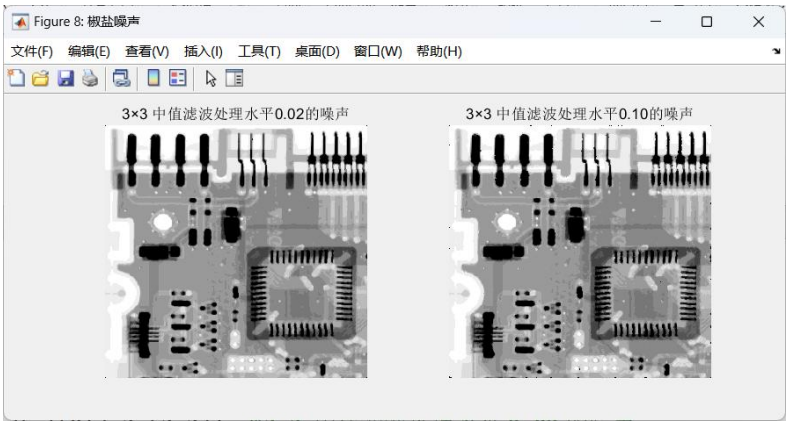



```
%-----------------------------------------------
J_3 = imnoise(I,'salt & pepper',0.02);%利用 imnoise 命令在图像 Sample2-1.jpg 上加入椒盐噪声
J_4 = imnoise(I,'salt & pepper',0.1);%加入更高水平的椒盐噪声
```

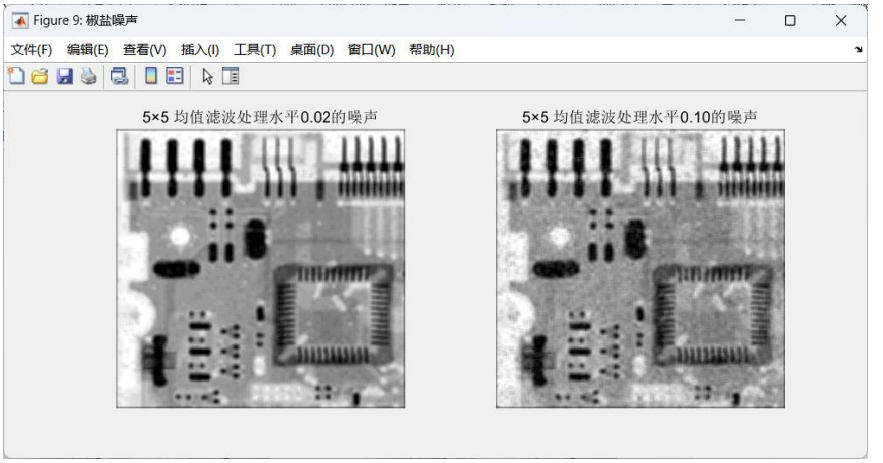
```
%3x3 模版下：
K_3 = filter2(ave_1,J_3)/255; %3×3 均值滤波处理水平 0.02 的噪声
K_4 = filter2(ave_1,J_4)/255; %3×3 均值滤波处理水平 0.10 的噪声

figure('Name','椒盐噪声')
subplot(1,2,1); imshow(K_3);title('3×3 均值滤波处理水平 0.02 的噪
声')
subplot(1,2,2); imshow(K_4);title('3×3 均值滤波处理水平 0.10 的噪
声')
```



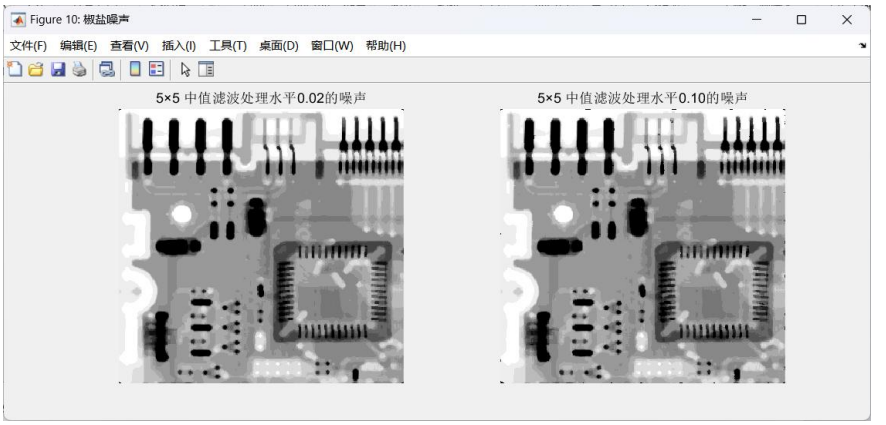

```
L_3 = medfilt2(J_3,[3 3]); %3×3 中值滤波处理水平 0.02 的噪声
L_4 = medfilt2(J_4,[3 3]); %3×3 中值滤波处理水平 0.10 的噪声

figure('Name','椒盐噪声')
subplot(1,2,1); imshow(L_3);title('3×3 中值滤波处理水平 0.02 的噪
声')
subplot(1,2,2); imshow(L_4);title('3×3 中值滤波处理水平 0.10 的噪
声')
```



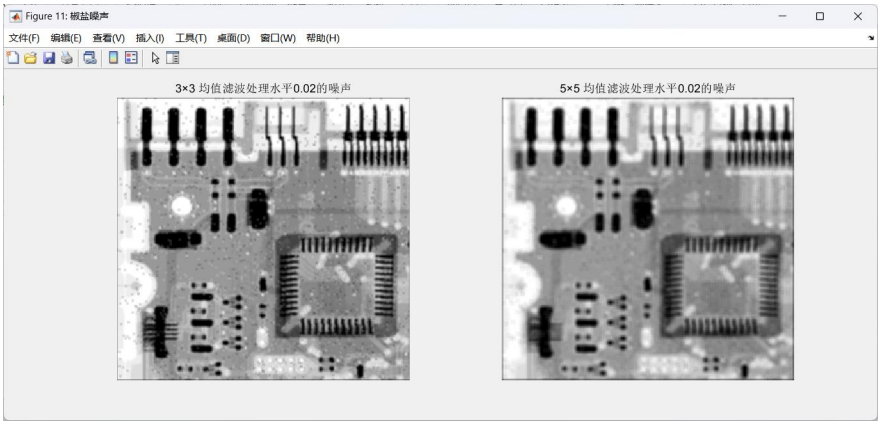
```
%5x5 模版下：
M_3 = filter2(ave_2,J_3)/255; %5x5 均值滤波处理水平 0.02 的噪声
M_4 = filter2(ave_2,J_4)/255; %5x5 均值滤波处理水平 0.10 的噪声

figure('Name','椒盐噪声')
subplot(1,2,1); imshow(M_3);title('5x5 均值滤波处理水平 0.02 的噪
声')
subplot(1,2,2); imshow(M_4);title('5x5 均值滤波处理水平 0.10 的噪
声')
```




```
N_3 = medfilt2(J_3,[5 5]); %5x5 中值滤波处理水平 0.02 的噪声
N_4 = medfilt2(J_4,[5 5]); %5x5 中值滤波处理水平 0.10 的噪声

figure('Name','椒盐噪声')
subplot(1,2,1); imshow(N_3);title('5x5 中值滤波处理水平 0.02 的噪
声')
subplot(1,2,2); imshow(N_4);title('5x5 中值滤波处理水平 0.10 的噪
声')
```



```
%对比 3x3 和 5x5 模版下同一噪声水平的处理
figure('Name','椒盐噪声')
subplot(1,2,1); imshow(K_3);title('3×3 均值滤波处理水平 0.02 的噪声')
subplot(1,2,2); imshow(M_3);title('5×5 均值滤波处理水平 0.02 的噪声')
```



讨论：

观察可知：

高斯噪声的干扰在图像上体现为模糊、椒盐噪声的干扰在图像上体现为噪点；

同水平下，椒盐噪声的干扰比高斯噪声的干扰更明显；

水平越高，噪声干扰体现在图片上越明显；

在同水平噪声下，模版越小，越能体现原始图像的清晰图；

四、 思考题

(1) 简述高斯噪声和椒盐噪声的特点。

高斯噪声的干扰在图像上体现为模糊、椒盐噪声的干扰在图像上体现为噪点；

(2) 结合实验内容，定性评价平均滤波器/中值滤波器对高斯噪声和椒盐噪声的去噪效果？

高斯噪声：

平均滤波器：去噪效果较好，但会模糊图像的边缘和细节；

中值滤波器：去噪效果较好，能够更好地保留图像的边缘和细节；

椒盐噪声：

平均滤波器：去噪效果较差，可能会扩散噪声并模糊图像；

中值滤波器：去噪效果非常好，能够有效去除噪声并保留图像的边缘和细节；

(3) 结合实验内容，定性评价滤波窗口对去噪效果的影响？
小窗口（如 3×3）
优点：
对图像的细节和边缘影响较小。小窗口的中值滤波器能够较好地保留图像的边缘和纹理，同时去除噪声。
对于椒盐噪声，小窗口的中值滤波器可以有效去除噪声像素，同时不会引入明显的伪影。
缺点：
对于高密度噪声，小窗口的中值滤波器可能无法完全去除噪声。噪声像素可能会在图像中残留；
大窗口（如 5×5）
优点：
对高密度噪声的去噪效果显著。大窗口的中值滤波器可以通过更大的邻域操作，更有效地去除噪声像素，尤其是在椒盐噪声的情况下。
缺点：
大窗口的中值滤波器可能会在图像的平滑区域引入“块状”伪影；
大窗口的中值滤波器需要对更多的像素进行排序和中值计算，计算时间会显著增加；
虽然中值滤波器比平均滤波器更能保留边缘，但大窗口仍可能导致边缘的轻微模糊

课堂实验三：图像增强—频域滤波

一、 实验目的

1. 掌握怎样利用傅立叶变换进行频域滤波
2. 掌握频域滤波的概念及方法
3. 熟练掌握频域空间的各类滤波器
4. 利用 MATLAB 程序进行频域滤波

二、 实验内容及步骤

1. 调入并显示所需的图片；
2. 利用 MATLAB 提供的低通滤波器实现图像信号的滤波运算，并与空间滤波进行比较。
3. 利用 MATLAB 提供的高通滤波器对图像进行处理。
4. 记录和整理实验报告。

三、 实验代码及结果

```
%1. 调入并显示所需的图片；
img_1=imread('room.tif');

% 显示原始图像
figure;
subplot(1, 3, 1);
imshow(img_1);
title('原始图像');
%------------------------------------------------------------
%2. 利用 MATLAB 提供的低通滤波器实现图像信号的滤波运算，并与空间滤波进行比较。
img_1_double = im2double(img_1);% 将图像转换为双精度浮点数
F = fft2(img_1_double);% 对图像进行傅里叶变换
[M, N] = size(F);% 获取图像大小

% 创建一个低通滤波器（理想低通滤波器）
D0 = 30; % 截止频率
[u, v] = meshgrid(1:N, 1:M);
u = u - floor(N/2);
```

```
v = v - floor(M/2);
D = sqrt(u.^2 + v.^2);
H = double(D <= D0);

H = ifftshift(H);% 将滤波器中心化
G = H .* F;% 应用滤波
img_filtered = ifft2(G);% 傅里叶逆变换
% 显示频率域低通滤波后的图像
subplot(1, 3, 2);
imshow(img_filtered, []);
title('频率域低通滤波后的图像');
%-------------------------------------------------------
% 使用均值滤波器进行空间滤波
h = fspecial('average', [3 3]); % 3x3 的均值滤波器
img_filtered_spatial = imfilter(img_1_double, h);

% 显示空间滤波后的图像
subplot(1, 3, 3);
imshow(img_filtered_spatial, []);
title('空间滤波后的图像');
```

频率域低通滤波后的图像特征：截止频率 30%。处理以后图像的高频成分被抑制，低频成分被保留，减少了噪声的同时还有减少大量细节，变得平滑。

空间滤波后的图像特征：均值滤波模板 3x3，处理以后图像的局部像素被平均处理，高频成分被抑制，不如原始图像清晰，与频率域低通滤波相比，模糊程度更小。边缘和细节保留度更高

```
%-------------------------------------------------------
%3. 利用 MATLAB 提供的高通滤波器对图像进行处理。
img_2=imread('number.tif');
```

```
figure;
subplot(1,3,1);
imshow(img_2, []);
title('原始');
%-----------------------------------------------
img_2_double = im2double(img_2);% 将图像转换为双精度浮点数
F = fft2(img_2_double);% 对图像进行傅里叶变换
[M, N] = size(F);% 获取图像大小
% 创建一个低通滤波器（理想低通滤波器）
D0 = 30; % 截止频率
[u, v] = meshgrid(1:N, 1:M);
u = u - floor(N/2);
v = v - floor(M/2);
D = sqrt(u.^2 + v.^2);
H = double(D > D0);

H = ifftshift(H);% 将滤波器中心化
G = H .* F;% 应用滤波
img_filtered = ifft2(G);% 傅里叶逆变换
% 显示频率域低通滤波后的图像
subplot(1,3,2);
imshow(img_filtered, []);
title('频率域高通滤波后的图像');
%-----------------------------------------------
% 使用均值滤波器进行空间滤波
h = fspecial('average', [3 3]); % 3x3 的均值滤波器
img_filtered_spatial = imfilter(img_2_double, h);

% 显示空间滤波后的图像
subplot(1, 3, 3);
imshow(img_filtered_spatial, []);
title('空间滤波后的图像');
```

频率域高通滤波后的图像特征：经过频率域高通滤波后，图像的低频成分被抑制，高频成分被保留，图像的边缘和细节被增强，背景变得模糊。

四、 思考题

1．结合实验，评价频域滤波有哪些优点？

频率域高通滤波优点：有效地增强了图像的边缘和细节，使图像看起来更加清晰。适合突出图像中的高频细节。

频率域低通滤波优点：有效地平滑了图像，减少了噪声，边缘变得柔和。适合去除图像中的高频噪声，使图像看起来更加平滑。

2．在频域滤波过程中需要注意哪些事项？

在应用滤波器之前，需要将滤波器中心化，以确保滤波器的频率响应与图像的频率分布对齐。

滤波器的设计需要根据具体的应用需求来确定。低通滤波器用于去除高频噪声，高通滤波器用于增强图像边缘。

课堂实验四：彩色图像处理

一、 实验目的

使用 MatLab 软件对图像进行彩色处理。使学生通过实验熟悉使用 MatLab 软件进行图像彩色处理的有关方法，并体会到图像彩色处理技术以及对图像处理的效果。

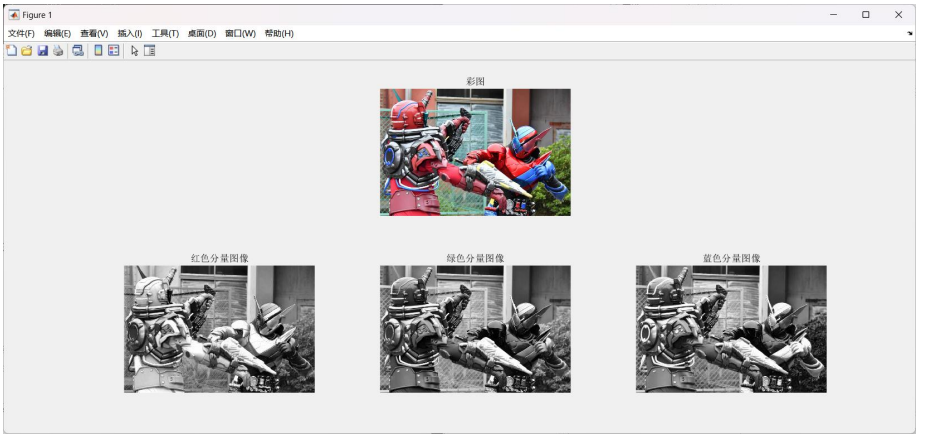
二、 实验内容及步骤

(1) 彩色图像的分析
调入并显示彩色图像 flower1.tif ；
拆分这幅图像，并分别显示其 R，G，B 分量；
根据各个分量图像的情况讨论该彩色图像的亮度、色调等性质。
(2) 彩色图像的直方图均衡
接内容(1)；
显示这幅图像的 R，G，B 分量的直方图，分别进行直方图均衡处理，并显示均衡后的直方图和直方图均衡处理后的各分量；
将处理完毕的各个分量合成彩色图像并显示其结果；
观察处理前后图像的彩色、亮度、色调等性质的变化。
(3) 假彩色处理
调入并显示红色可见光的灰度图像 vl_red.jpg、绿色可见光的灰度图像 vl_green.jpg 和蓝色可见光的灰度图像 vl_blue.jpg;以及近红外灰度图像 infer_near.jpg 和中红外灰度图像 infer_mid.jpg；
以图像 vl_red.jpg 为 R；图像 vl_green.jpg 为 G；图像 vl_blue.jpg 为 B，将这三幅图像组合成可见光 RGB 彩色图像；
分别以近红外图像 infer_near.jpg 和中红外图像 infer_mid 替换 R 分量，形成假彩色图像；
观察处理的结果，注意不同波长红外线图像组成图像的不同结果 14
(4) 伪彩色处理 1：灰度切片处理
调入并显示灰度图像 head.jpg；
利用 MATLAB 提供的函数对图像在 8~256 级的范围内进行切片处理，并使用 hot 模式和 cool 模式进行彩色化；
观察处理的结果。
(5) 彩色变换(选做)
调入并显示灰度图像 Lenna.jpg；
使用不同相位的正弦函数作为变换函数，将灰度图像变换为 RGB 图像。

其中红色分量 R 的变换函数为-sin( )，绿色分量 G 的变换函数为-cos( );，蓝色分量 B 的变换函数为 sin( )；
显示变换曲线及变换合成的彩色图像并观察彩色变换图像的色调与原始图像灰度之间的关系；
将 RGB 的变换公式至少互换一次(例如 R 与 G 互换)，显示变换曲线、变换结果并观察处理的结果。
(6)打印全部结果并进行讨论。

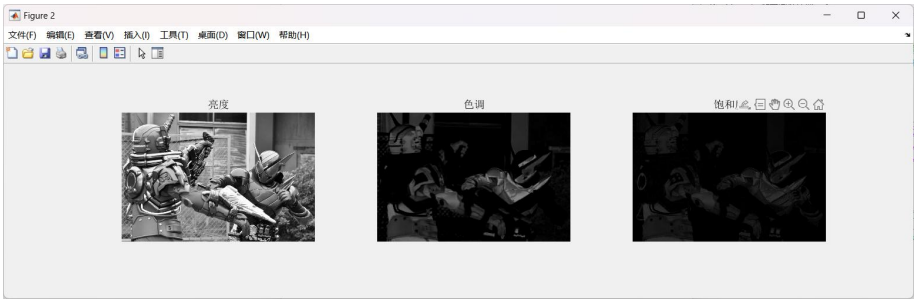
三、 实验代码及结果

```
%调入并显示彩色图像 flower1.tif ，拆分这幅图像，并分别显示其 R，G，B 分量;
rgb_image=imread('flowerl.tif'); %读取图像 flower1.tif
fR=rgb_image(:,:,1); %获取图像的红色分量
fG=rgb_image(:,:,2); %获取图像的绿色分量
fB=rgb_image(:,:,3); %获取图像的蓝色分量
figure(1),
subplot(2,3,2);imshow(rgb_image);title('彩图')%分别显示图像
subplot(2,3,4);imshow(fR);title('红色分量图像')
subplot(2,3,5);imshow(fG);title('绿色分量图像')
subplot(2,3,6);imshow(fB);title('蓝色分量图像')
```



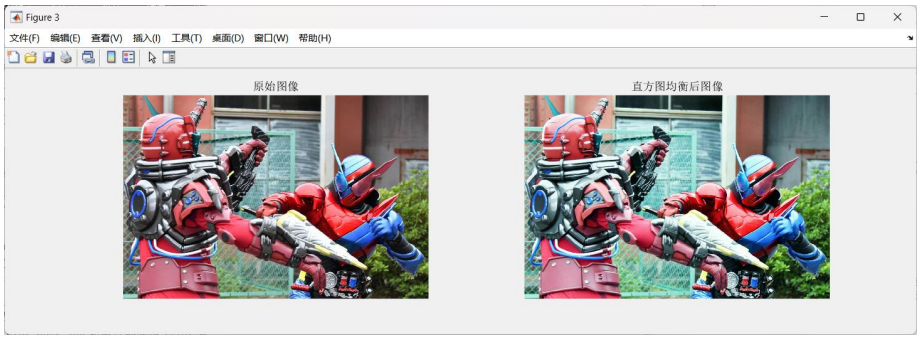

```
%rgb 图像转化为 NTSC 彩色空间的图像
yiq_image=rgb2ntsc(rgb_image);
fY=yiq_image(:,:,1); %图像 flower1.tif 的亮度
fI=yiq_image(:,:,2); %图像 flower1.tif 的色调
fQ=yiq_image(:,:,3); %图像 flower1.tif 的饱和度
figure(2),
subplot(1,3,1);imshow(fY);title('亮度')
```

```
subplot(1,3,2);imshow(fI);title('色调')
subplot(1,3,3);imshow(fQ);title('饱和度')
```



观察发现，原始彩色图像的亮度较高，因为红色分量图像的亮度较高。
原始彩色图像的色调色调丰富，大部分是红绿蓝三色。
原始彩色图像的饱和度高。

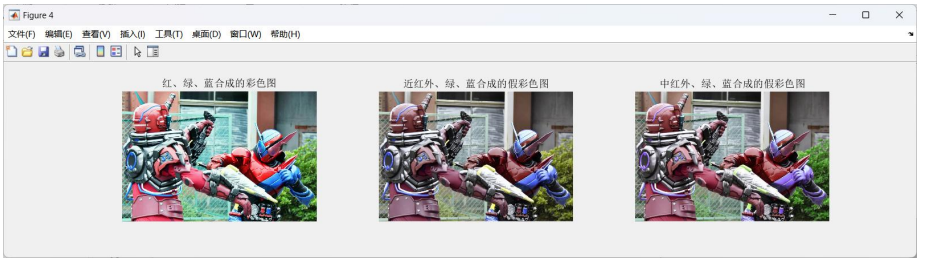
```
%-------------------------------------------------------------
%彩色图像的直方图均衡
fR=histeq(fR,256); %对彩色图像的分量进行直方图均衡化
fG=histeq(fG,256);
fB=histeq(fB,256);
RGB_image=cat(3,fR,fG,fB); %将直方图均衡化后的彩色图像合并
figure(3),
subplot(1,2,1);imshow(rgb_image);title('原始图像')
subplot(1,2,2);imshow(RGB_image);title('直方图均衡后图像')
```



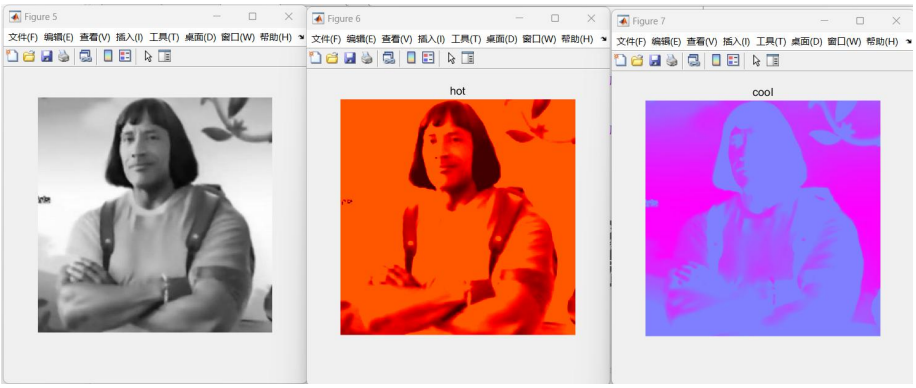
讨论：直方图均衡后图像亮度明显更亮,色调整体都变浅了一些。

```
%-----------------------------------------
%假彩色处理
f1=imread('vl_red.jpg');
f2=imread('vl_green.jpg');
f3=imread('vl_blue.jpg');
f4=imread('infer_near.jpg');
f5=imread('infer_mid.jpg');
```

```
ture_color=cat(3,fR,fG,fB);
figure(4),
subplot(1,3,1);imshow(ture_color);title('红、绿、蓝合成的彩色图
') %显示由红、绿、蓝三幅图合成的彩色图

false_color_1=cat(3,f4,f2,f3); %用近红外图像代替 R 分量
subplot(1,3,2);imshow(false_color_1);title('近红外、绿、蓝合成的
假彩色图')%显示由近红外、绿、蓝三幅图合成的假彩色图

false_color_2=cat(3,f5,f2,f3); %用种红外图像代替 R 分量
subplot(1,3,3);imshow(false_color_2);title('中红外、绿、蓝合成的
假彩色图')%显示由中红外、绿、蓝三幅图合成的假彩色图
```

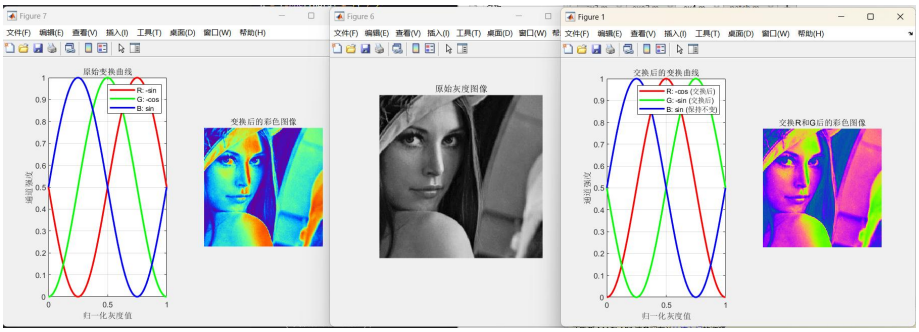



```
%-------------------------------------------
%伪彩色处理 1：灰度切片处理
f1=imread('head.jpg');
f2=rgb2gray(f1);%彩图转灰度图
figure(5),imshow(f2);
cut_1=imadjust(f2,[0.0925 0.5],[0.0925 0.5]);%提取灰度在 16-128
之间的像素
cut_2=imadjust(f2,[0.5 1],[0.5 1]); %提取灰度在 128-256 之间的像
素
figure(6),imshow(cut_1),colormap(hot),title('hot') ;%显示图像
cut_1,并使用 hot 模型彩色化
figure(7),imshow(cut_2),colormap(cool),title('cool') ; %显示图
像 cut_2,并使用 cool 模型彩色化
```




```
%-----------------------
%彩色变换(选做)
% 读取灰度图像
% 读取并显示灰度图像
img_gray = imread('Lenna.jpg');
img = im2double(img_gray);
figure(6);
imshow(img);
title('原始灰度图像');
% 生成原始变换的 RGB 图像
theta = img * 2 * pi;
R = (-sin(theta) + 1) / 2;
G = (-cos(theta) + 1) / 2;
B = (sin(theta) + 1) / 2;
rgb_img = cat(3, R, G, B);

% 显示原始变换曲线
x = linspace(0, 1, 1000);
theta_x = x * 2 * pi;
R_curve = (-sin(theta_x) + 1) / 2;
G_curve = (-cos(theta_x) + 1) / 2;
B_curve = (sin(theta_x) + 1) / 2;
figure(7);
subplot(1,2,1);
plot(x, R_curve, 'r', 'LineWidth', 2);
hold on;
plot(x, G_curve, 'g', 'LineWidth', 2);
plot(x, B_curve, 'b', 'LineWidth', 2);
```

```
legend('R: -sin', 'G: -cos', 'B: sin');
title('原始变换曲线');
xlabel('归一化灰度值');
ylabel('通道强度');
grid on;
% 显示变换后的彩色图像
subplot(1,2,2),
imshow(rgb_img);
title('变换后的彩色图像');

% 互换 R 和 G 的变换函数
R_swapped = (-cos(theta) + 1) / 2;
G_swapped = (-sin(theta) + 1) / 2;
B_swapped = B;
rgb_swapped = cat(3, R_swapped, G_swapped, B_swapped);
% 显示交换后的变换曲线
R_swapped_curve = (-cos(theta_x) + 1) / 2;
G_swapped_curve = (-sin(theta_x) + 1) / 2;
B_swapped_curve = B_curve;

figure;
subplot(1,2,1);
plot(x, R_swapped_curve, 'r', 'LineWidth', 2);
hold on;
plot(x, G_swapped_curve, 'g', 'LineWidth', 2);
plot(x, B_swapped_curve, 'b', 'LineWidth', 2);
legend('R: -cos (交换后)', 'G: -sin (交换后)', 'B: sin (保持不变)');
title('交换后的变换曲线');
xlabel('归一化灰度值');
ylabel('通道强度');
grid on;

% 显示交换后的彩色图像
subplot(1,2,2);
imshow(rgb_swapped);
title('交换 R 和 G 后的彩色图像');
```



四、 思考题

1. 为什么经彩色直方图均衡后的图像除了对比度会有所增强外，还有色调的变化？

RGB 颜色空间的非线性特性使得直接操作会导致颜色的非线性变化。直方图均衡化基于灰度值操作，没有考虑颜色之间的相互关系。在 RGB 颜色空间中，直接对每个通道进行均衡化会破坏颜色的一致性，直方图均衡化对每个颜色通道独立进行，改变了颜色之间的相对关系

2. 实验内容(3)的假彩色处理方案是否可以有多种？若有，请估计其它方案的可能结果。

有，可以让近红外/中红外替换 G/B 分量。

近红外替换 G 分量：绿色更鲜艳

中红外替换 B 分量：蓝色更鲜艳

3. 在实验内容(4)中，对于灰度切片处理的图像 head.gif 使用多少级切片比较合适？

方案 1：head.gif 的细节较少，使用 8 到 16 级切片。这些等级可以在保留基本特征的同时，避免过多的细节干扰。

方案 2：head.gif 的细节适中，使用 32 到 64 级切片。这些等级可以更好地保留图像的细节，同时突出主要特征。

方案 3：head.gif 的细节丰富，使用 128 到 256 级切片。这些等级可以更细致地保留图像的细节，适合需要高分辨率分析的场景。