|  |
| --- |
| **数字图像处理实验报告** |
| **实验一：数字图像处理的基础知识、数字变换与图像增强** |
| |  |  | | --- | --- | | 姓　 名： |  | | 学 科 专 业： |  | | 指 导 教 师： |  | | 班 级： |  | | 学 号： |  | |
| 二○二二年十一月 |

**目录**

[一、实验要求 1](#_Toc28819)

[二、实验目的 1](#_Toc11875)

[三、实验内容 1](#_Toc4168)

[2.1 第一题 1](#_Toc4046)

[2.1.1 题目 1](#_Toc31760)

[2.1.2 实验结果 1](#_Toc10107)

[2.2 第二题 2](#_Toc658)

[2.2.1 题目 2](#_Toc3958)

[2.2.2 实验结果 2](#_Toc11200)

[2.3 第三题 2](#_Toc30128)

[2.3.1 题目 2](#_Toc17513)

[2.3.2 实验结果 3](#_Toc30679)

[2.4 第四题 3](#_Toc20272)

[2.4.1 题目 3](#_Toc17130)

[2.4.2 实验结果 3](#_Toc20063)

[2.5 第五题 5](#_Toc15428)

[2.5.1 题目 5](#_Toc13619)

[2.5.2 实验结果 5](#_Toc26685)

[四、 实验代码 7](#_Toc1164)

# 

# 实验要求

（1）数字图像的读取与保存；

（2）图像彩色空间的转换；

（3）图像几何变换与插值；

（4）图像的平滑与锐化

# 二、实验目的

掌握数字图像处理的基础知识，图像的色彩空间与转换，图像的空间几何变换，傅里叶、离散余弦和小波变换，正确使用图像质量的评价指标；基本掌握图像增强方法，了解图像退化模型及模型参数估计，掌握图像复原方法；

实验形式：独立完成

实验考核：提交实验报告。根据设计方案、实验结果、及实验报告质量综合评定成绩。

# 三、实验内容

## 2.1 第一题

### 2.1.1 题目

读入一幅 RGB 图像，变换为灰度图像和二值图像，并在同一个窗口内分成三个子窗口来分别显示 RGB 图像和灰度图像，注上文字标题。

### 2.1.2 实验结果

实验结果如图1所示，左图为原始图像，中图为灰度图像，右图为二值化后的图像。由图我们可以看出二值化的效果可能并不是很好，后续可以尝试使用大津法进行二值化，或许会有更好的二值化效果。

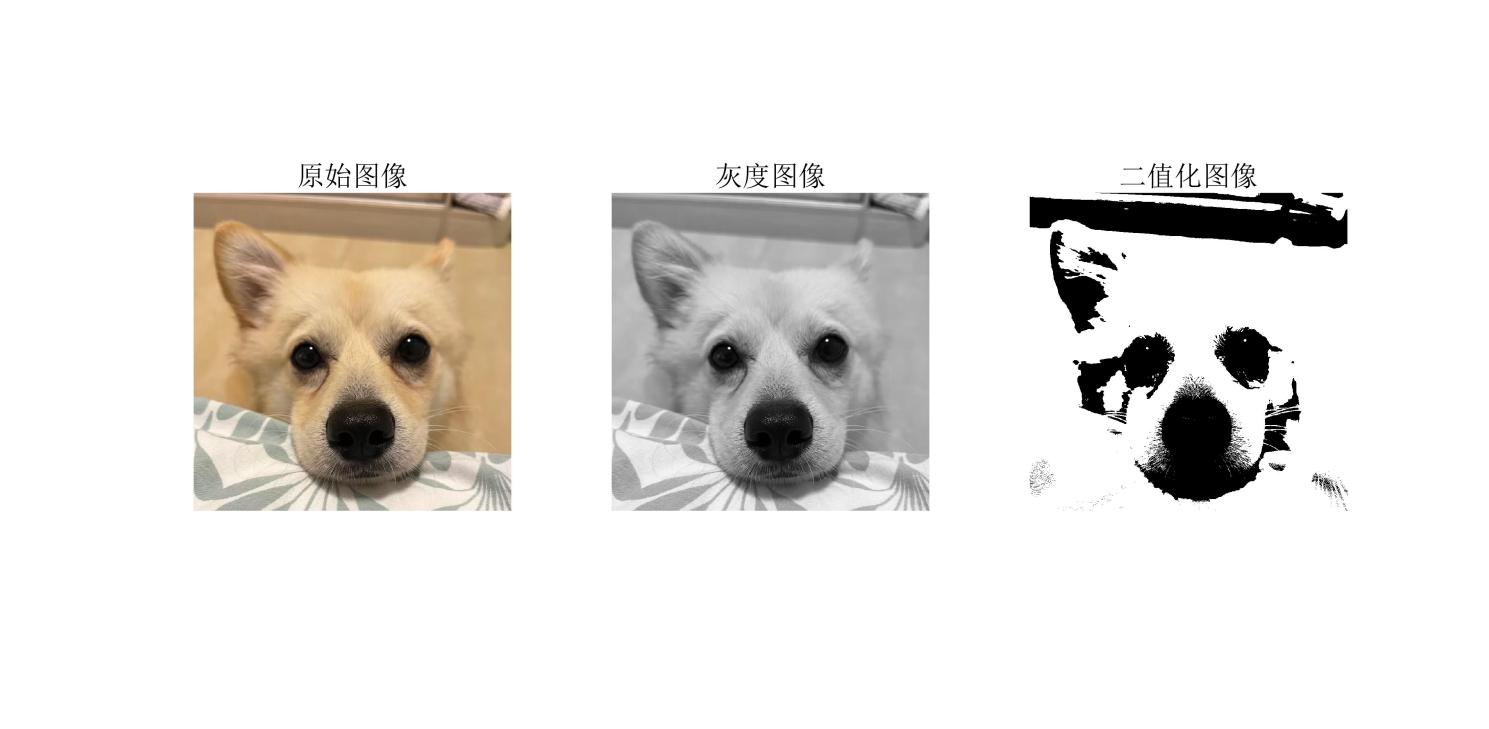


图 1 第一题实验结果

## 2.2 第二题

### 2.2.1 题目

读入一幅灰度图像，对其进行傅里叶变换和一级小波分解变换，并在同一个窗口内分成三个子窗口来分别显示原始灰度图像、傅里叶变换和一级小波分解变换的图像，注上文字标题。

### 2.2.2 实验结果

图像经过傅里叶变换和小波变换的图像如图2所示。

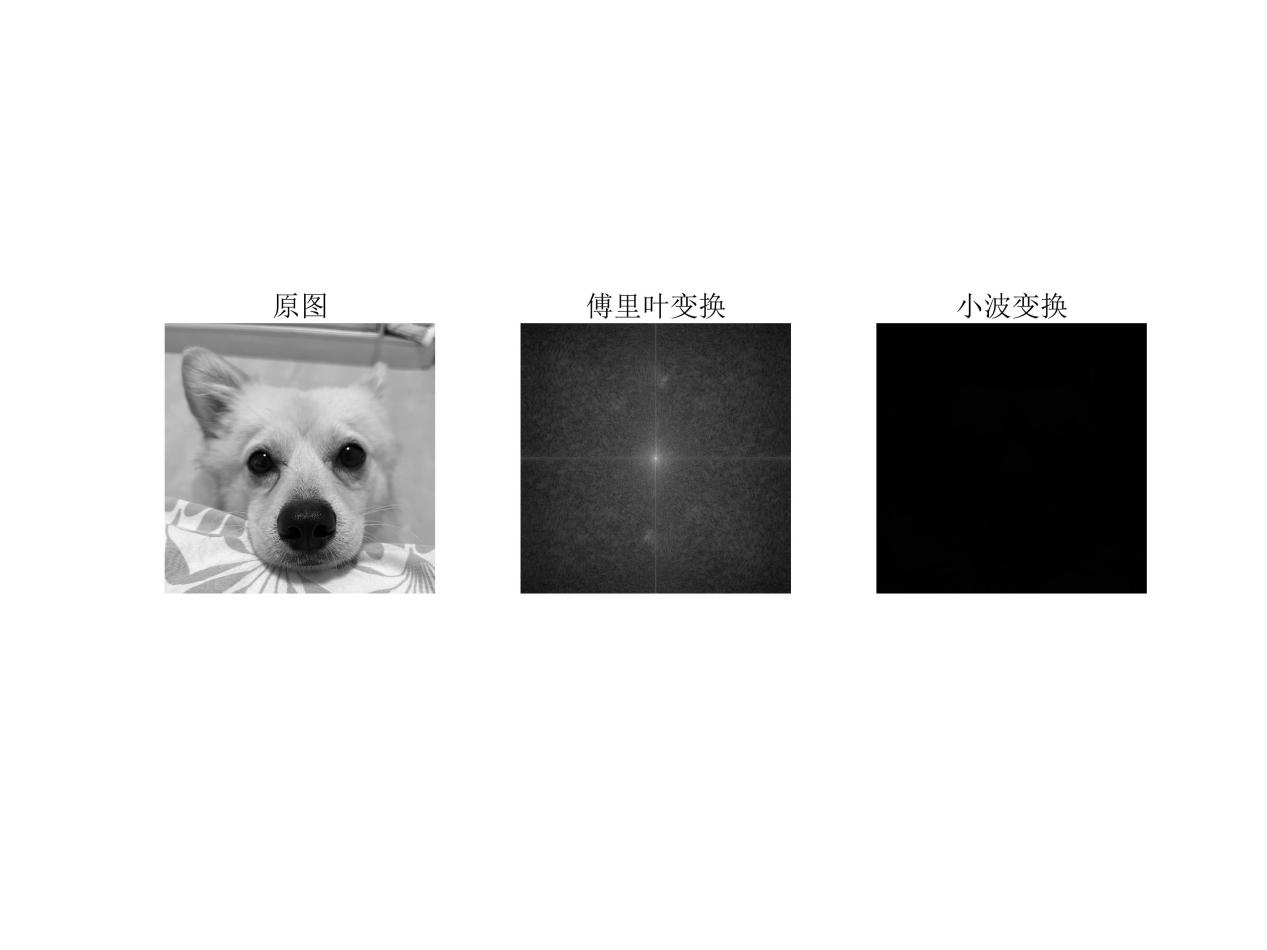


图 2 第二题实验结果

## 2.3 第三题

### 2.3.1 题目

图像向上、向右平移，并用白色填充空白部分，再对其做水平镜像，然后旋转 45 度，再缩小 4 倍。最后，在同一个窗口内分成两个子窗口来分别显示原图像和经历复合变换后的图像，注上文字标题。

### 2.3.2 实验结果

由图3我们可以看出，图片向上、向右平移后，使用白色填充了平移部分。再对这部分图像进行水平镜像并旋转45度和缩小四倍得到了符合变换后的图像。

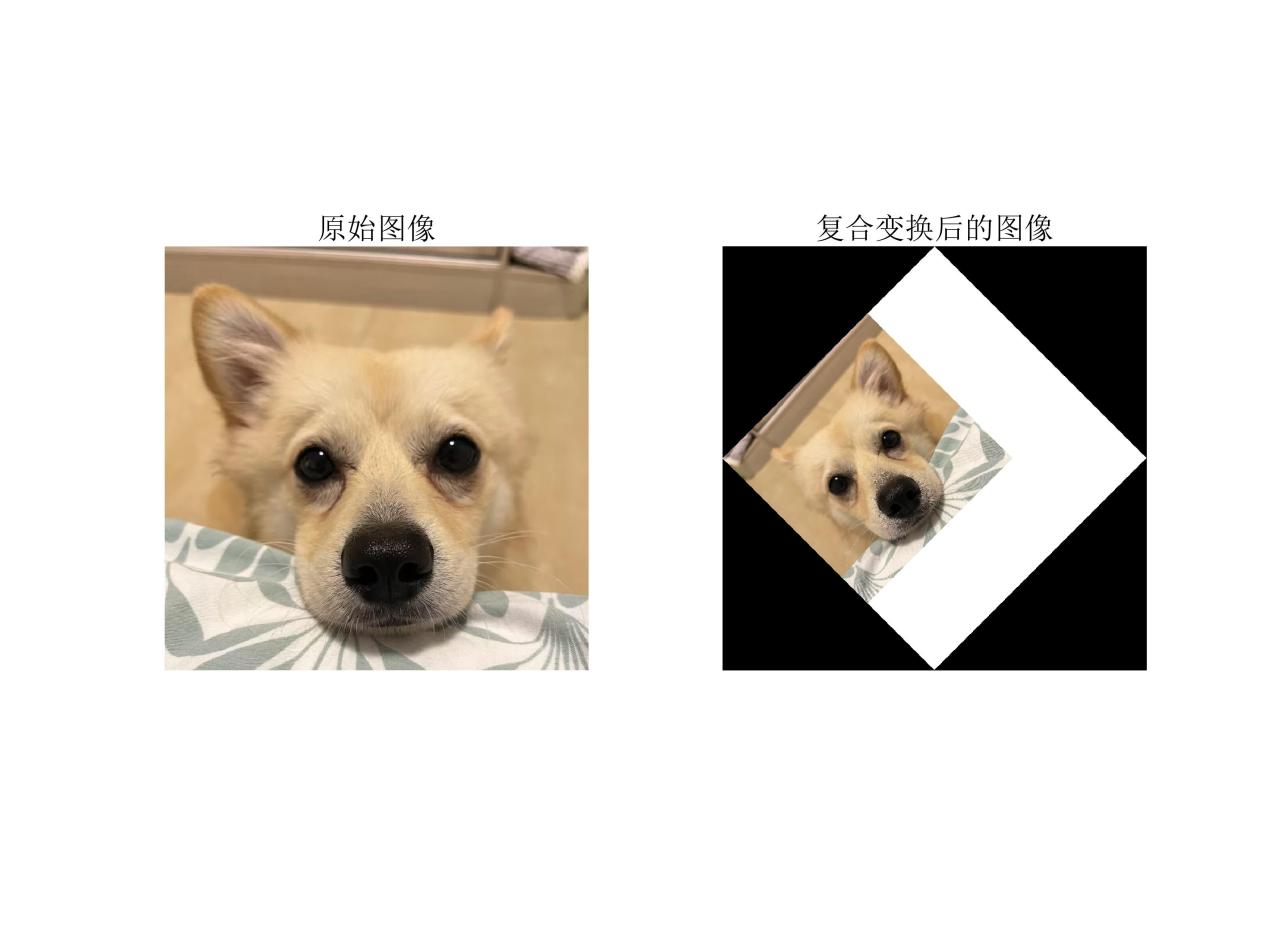


图 3 第三题实验结果

## 2.4 第四题

### 2.4.1 题目

加入高斯噪声和乘性噪声，输出图像；选择合适的方法分别对高斯和乘性噪声去噪，然后输出去噪图像。

### 2.4.2 实验结果

高斯噪声是指它的[概率密度函数](http://baike.baidu.com/view/998459.htm" \o "概率密度函数)服从[高斯分布](http://baike.baidu.com/view/573667.htm" \o "高斯分布)（即[正态分布](http://baike.baidu.com/view/45379.htm" \o "正态分布)）的一类噪声。如果一个噪声，它的幅度分布服从[高斯分布](https://so.csdn.net/so/search?q=%E9%AB%98%E6%96%AF%E5%88%86%E5%B8%83&spm=1001.2101.3001.7020" \o "高斯分布)，而它的[功率谱密度](http://baike.baidu.com/view/1230146.htm" \o "功率谱密度)又是均匀分布的，则称它为高斯白噪声。高斯白噪声的二阶矩不相关，一阶矩为[常数](http://baike.baidu.com/view/122755.htm" \o "常数)，是指先后信号在时间上的相关性。产生原因：1.图像传感器在拍摄时市场不够明亮、亮度不够均匀；2.电路各元器件自身噪声和相互影响；3.图像传感器长期工作，温度过高。

乘性噪声一般由信道不理想引起，它们与信号的关系是相乘，信号在它在，信号不在他也就不在。

加入高斯噪声和乘性噪声的图片如图4所示。

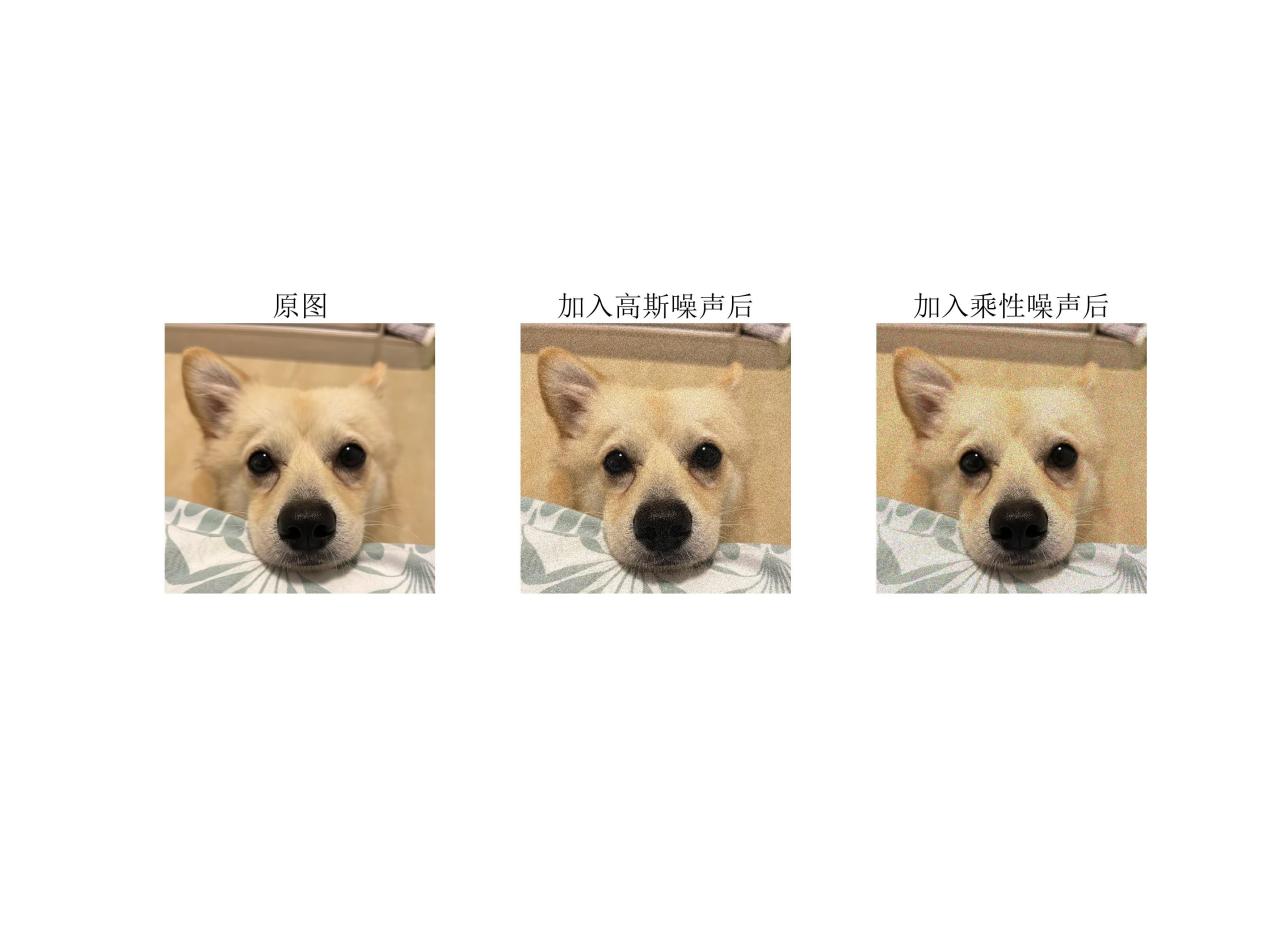


图 4 对图像分别加入高斯噪声和乘性噪声

对于加入高斯噪声和乘性噪声噪声的图片分别使用均值滤波、中值滤波、二次维纳滤波、高斯滤波对加入噪声后的图片进行去噪。

结果如图5、图6所示，从结果我们可以看出二次维纳滤波对高斯噪声的图像滤波效果好，它比线性滤波器具有更好的选择性，可以更好地保存图像的边缘和高频细节信息。均值滤波对乘性噪声的去噪效果较好。

总的来说，均值滤波和二次维纳滤波对于两种噪声的去噪效果相对于其他两种滤波算法更好。

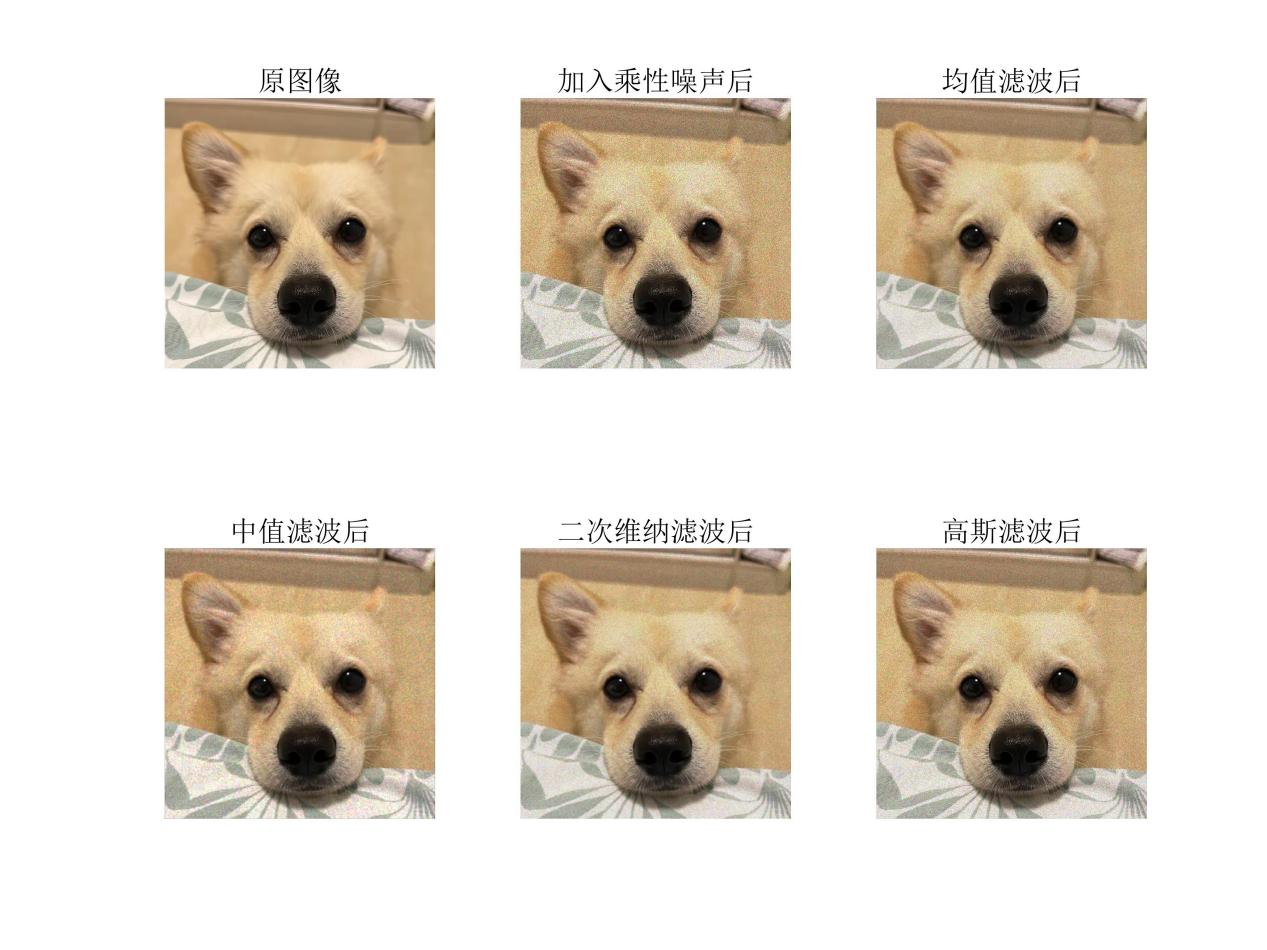


图 5 对加入乘性噪声的图片进行去噪

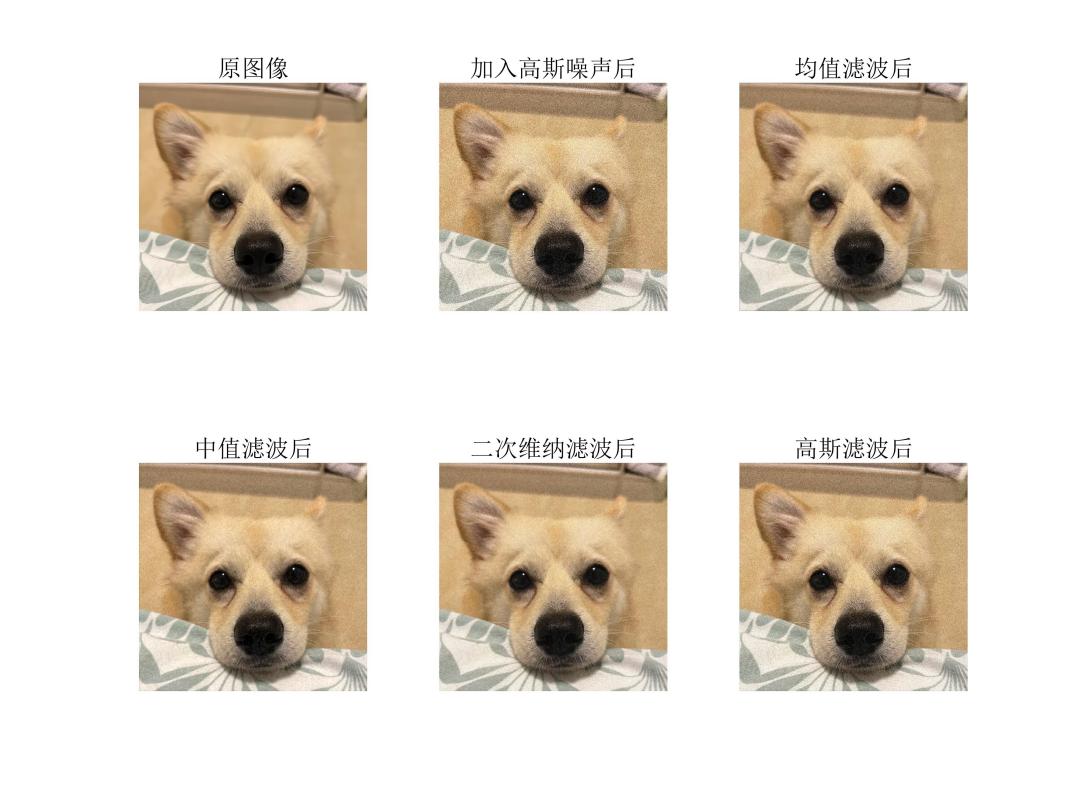


图 6 对加入高斯噪声的图片进行去噪

## 2.5 第五题

### 2.5.1 题目

采用三种不同算子对图像进行锐化处理，在同一窗口输出图像；使用同态滤波器实现图像增强，输出图像。

### 2.5.2 实验结果

分别使用Roberts算子、Laplace算子、Sobel算子、Prewitt算子对图像进行锐化处理，处理结果如图7所示。

同态滤波利用去除乘性噪声，可以同时增加对比度以及标准化亮度，借此达到图像增强的目的。

一副图像可以表示为其照度分量和反射分量的乘积，虽然在时域上这两者是不可分离的，但是经由傅立叶转换两者在频域中可以线性分离。由于照度可视为环境中的照明，相对变化很小，可以看作是图像的低频成分；而反射率相对变化较大，则可视为高频成分。通过分别处理照度和反射率对像元灰度值的影响，通常是借由高通滤波器，让图像的照明更加均匀，达到增强阴影区细节特征的目的。

使用同态滤波器对图片进行增强，增强结果如图8所示。

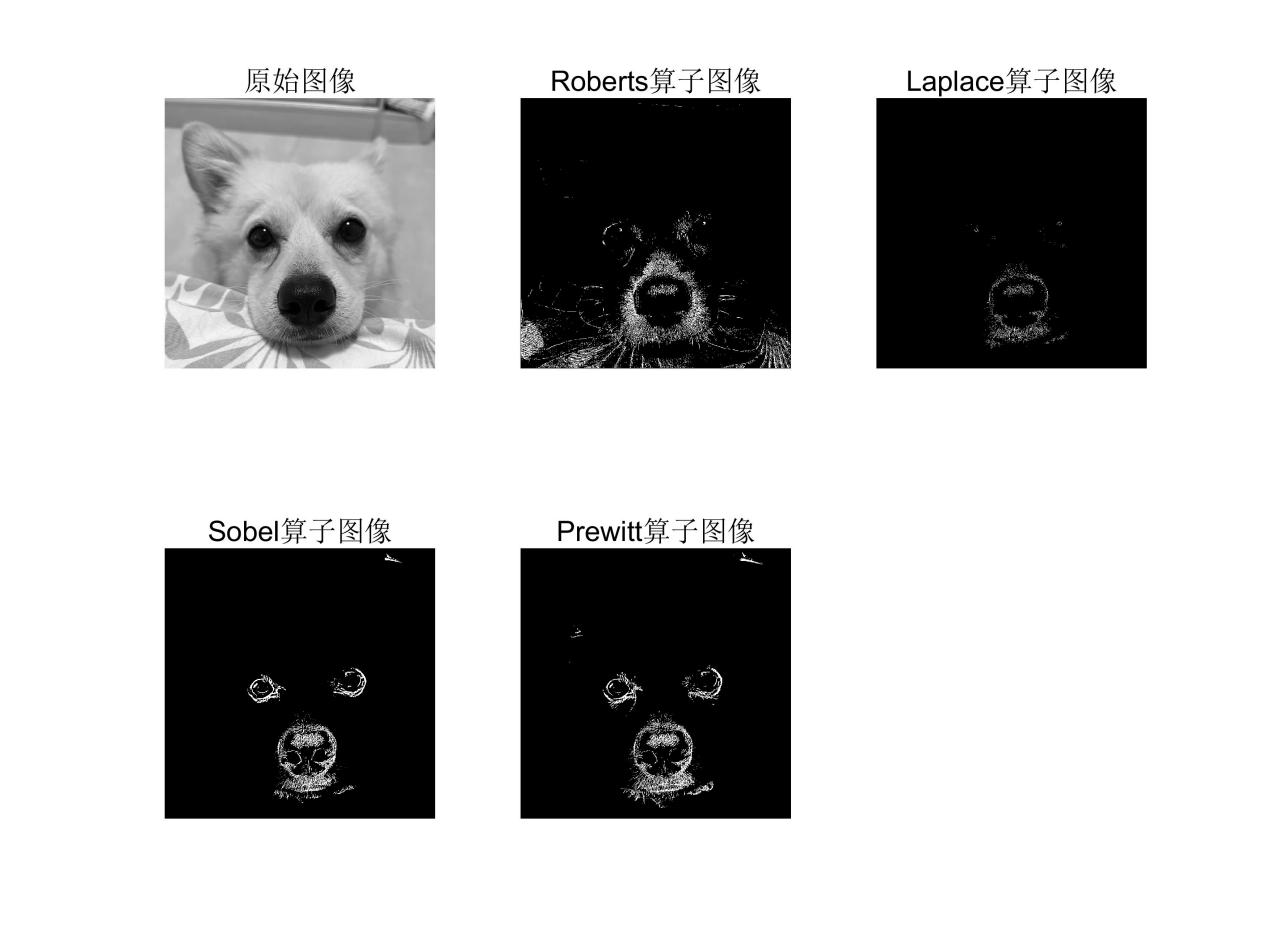


图 7 使用不同算子对图像进行锐化处理



图 8 同态滤波结果图

# 实验代码

%% 第一题

%读入一幅 RGB 图像，变换为灰度图像和二值图像，并在同一个窗口内分成三个子窗口来分别显示 RGB 图像和灰度图像，注上文字标题。

% 读入RGB图像

figure(1)

init = imread('51.png');

subplot(1,3,1)

imshow(init)

title('原始图像')

%转换为灰度图

gray = rgb2gray(init);

subplot(1,3,2)

imshow(gray)

title('灰度图像')

%二值化图像

thresh = graythresh(gray); %自动确定二值化阈值

BW = imbinarize(gray,thresh); %对图像二值化

subplot(1,3,3)

imshow(BW)

title('二值化图像')

%% 第三题

%读入一幅 RGB 图像，编写 MATLAB 程序实现下列变换：图像向上、向右平移，并用白色填充空白部分，

%再对其做水平镜像，然后旋转 45 度，再缩小 4 倍。

%最后，在同一个窗口内分成两个子窗口来分别显示原图像和经历复合变换后的图像，注上文字标题。

figure(2)

init2 = imread('51.png');

% se = translate(strel(1), [-250 250]);%将一个平面结构化元素分别向下和向右移动30个位置

% J = imdilate(init2,se);%利用膨胀函数平移图像

J = move(init2,-500,500);

J = mirror(J,0);

J = imrotate(J,45);

aa = 0.25;

J = imresize(J,aa,'nearest');

figure(2)

subplot(1,2,1)

imshow(init2)

title('原始图像')

subplot(1,2,2)

imshow(J)

title('复合变换后的图像');

%% 第五题

%读入一幅 RGB 图像，编写 MATLAB 程序实现下列要求：

%采用三种不同算子对图像进行锐化处理，在同一窗口输出图像；使用同态滤波器实现图像增强，输出图像。

figure(3)

img = imread('51.png');

img = rgb2gray(img);

subplot(2,3,1);

imshow(img);

title('原始图像');

[m,n] = size(img);

T = 20;

%Roberts算子

imgr = zeros(m,n);

for i=2:m-1

for j=2:n-1

imgr(i,j)= abs(img(i,j)-img(i+1,j+1)) + abs(img(i+1,j)-img(i,j+1));

if imgr(i,j)<T

imgr(i,j) = 0;

else

imgr(i,j) = 255;

end

end

end

subplot(2,3,2);

imshow(imgr);

title('Roberts算子图像');

%Laplace算子

imgl = zeros(m,n);

for i=2:m-1

for j=2:n-1

imgl(i,j)= abs(img(i+1,j)+img(i-1,j)+img(i,j+1)+img(i,j-1)-4\*img(i,j));

if imgl(i,j)<T

imgl(i,j) = 0;

else

imgl(i,j) = 255;

end

end

end

subplot(2,3,3);

imshow(imgl);

title('Laplace算子图像');

%Sobel算子

imgs = zeros(m,n);

for i=2:m-1

for j=2:n-1

imgs(i,j)= abs(img(i-1,j+1)+2\*img(i,j+1)+img(i+1,j+1)-img(i-1,j-1)-2\*img(i,j-1)-img(i+1,j-1)) + abs(img(i+1,j-1)+2\*img(i+1,j)+img(i+1,j+1)-img(i-1,j-1)-2\*img(i-1,j)-img(i-1,j+1));

if imgs(i,j)<T

imgs(i,j) = 0;

else

imgs(i,j) = 255;

end

end

end

subplot(2,3,4);

imshow(imgs);

title('Sobel算子图像');

%Prewitt算子

imgp = zeros(m,n);

for i=2:m-1

for j=2:n-1

imgp(i,j)= abs(img(i-1,j+1)+img(i,j+1)+img(i+1,j+1)-img(i-1,j-1)-img(i,j-1)-img(i+1,j-1)) + abs(img(i+1,j-1)+img(i+1,j)+img(i+1,j+1)-img(i-1,j-1)-img(i-1,j)-img(i-1,j+1));

if imgp(i,j)<T

imgp(i,j) = 0;

else

imgp(i,j) = 255;

end

end

end

subplot(2,3,5);

imshow(imgp);

title('Prewitt算子图像');

%% 同态滤波

In=imread('51.png');

I=rgb2gray(In);

[H,W]=size(I);

rH=2;

rL=0.2;

C=2;

D0=max(H,W);

J=homomorphicfiltering(I,rH,rL,C,D0);

%figure(4)

% subplot(2,2,1);imshow(I);title('原始图像');

% subplot(2,2,2);imshow(J);title('同态滤波后的图像');

% subplot(2,2,3);imhist(I);title('原始图像直方图');

% subplot(2,2,4);imhist(J);title('同态滤波后的图像直方图');

figure(4)

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

subplot(1,2,2);imshow(J);title('同态滤波后的图像');

function J = mirror(I, a)

[M, N, G] = size(I);

I = im2double(I);

J = ones(M, N, G);

for i = 1:M

for j = 1:N

if(a == 0)%水平镜像

J(i, N-j+1, :) = I(i, j, :);

elseif(a == 1 )%垂直镜像

J(M-i+1, j, :) = I(i, j, :);

elseif(a == 2 )%水平垂直

J(M-i+1, N-j+1, :) = I(i, j, :);

end

end

end

function J = move(I, a, b)

[M, N, G] = size(I);

I = im2double(I);

J = ones(M + abs(a), N + abs(b), G); %建立新的矩阵，将新图像扩大，避免越过边界

for i = 1:M

for j = 1:N

if(a < 0 && b < 0)

J(i, j, :) = I(i, j, :);

elseif(a > 0 && b > 0)

J(i + a, j + b, :) = I(i, j , :);

elseif(a < 0 && b > 0)

J(i, j + b, :) = I(i, j, :);

else

J(i + a, j, :) = I(i, j, :);

end

end

end

end

% 同态滤波器

% ImageIn - 进行滤波的输入灰度图像(若输入RGB图像需要先利用rgb2gray函数转换成灰度图像)

% High - 高频增益,取值大于1

% Low - 低频增益,取值在0和1之间

% C - 锐化系数

% D0 - 截止频率(越大图像越亮)

% ImageOut - 滤波后的输出灰度图像

function [ImageOut] = homomorphicfiltering(ImageIn, High, Low, C, D0)

I = double(ImageIn); % 将图像类型转换为双精度型,不会改变数据本身便于进行傅里叶变换

[M,N] = size(ImageIn); % 返回输入图像行数和列数

a = floor(M / 2); % 中心点坐标

b = floor(N / 2);

LogImg = log(I + 1); % 对输入图像取对数

Log\_FFT = fft2(LogImg);

D=zeros(M,N); % 初始化中间变量D和H

H=zeros(M,N);

for i = 1 : M

for j = 1 : N

D(i,j)=sqrt((i - a)^2 + (j - b)^2); % 频率点(i,j)到频率中心的距离

H(i, j)=(High - Low) \* (1 - exp(-C \* (D(i,j)^2 / (D0 ^2)))) + Low; % 同态滤波器函数

end

end

H = ifftshift(H); % 对H做反中心化

Log\_FFT = H.\* Log\_FFT; % 滤波，矩阵点乘

Log\_FFT = ifft2(Log\_FFT); % 反傅立叶变换

ImageOut = real(exp(Log\_FFT)-1); % 取指数

end

I=imread('51.png');

I=rgb2gray(I);

I=im2double(I);%将图像的数据格式转换为double型

F=fft2(I);%进行傅里叶变换

F=fftshift(F);

F=abs(F);%求傅里叶变换的模

T=log(F+1);。

[A,H,V,D]=dwt2(I,'db2');

M=figure(1);

subplot(131),imshow(I),title('原图');

subplot(132),imshow(T,[]),title('傅里叶变换');

subplot(133),imshow(uint8(A)),title('小波变换');

I=imread('51.png');

[m,n,z]=size(I);

y=0+0.1\*randn(m,n);%二维高斯分布矩阵 0是均值 0.1是标准差

%先将其double化，再除以255 便于后面计算

I1=double(I)/255;

%加上噪声

I1=I1+y;

%将像素范围扩大至0--255

I1=I1\*255;

%转换为uint8类型

I1=uint8(I1);

I2=imnoise(I,'speckle');%加入乘性噪声

figure(1);

subplot(1,3,1),imshow(I),title('原图');

subplot(1,3,2),imshow(I1),title('加入高斯噪声后');

subplot(1,3,3),imshow(I2),title('加入乘性噪声后');

R\_1 = I2(:,:,1);

G\_1 = I2(:,:,2);

B\_1 = I2(:,:,3);

h=fspecial('average',3);

%fspecial用于产生预定义滤波器

R1\_junzhi=uint8(round(filter2(h,R\_1)));

G1\_junzhi=uint8(round(filter2(h,G\_1)));

B1\_junzhi=uint8(round(filter2(h,B\_1)));

%均值滤波，uin8为变量类型，round为取整，h为滤波参数即均值，h后的变量为要处理的图像

R1\_zhongzhi=medfilt2(R\_1,[5,5]);

G1\_zhongzhi=medfilt2(G\_1,[5,5]);

B1\_zhongzhi=medfilt2(B\_1,[5,5]);

%中值滤波

R1\_wiener1=wiener2(R\_1,[5,5]);

G1\_wiener1=wiener2(G\_1,[5,5]);

B1\_wiener1=wiener2(B\_1,[5,5]);

%第一次维纳滤波

R1\_wiener2=wiener2(R\_1,[5,5]);

G1\_wiener2=wiener2(G\_1,[5,5]);

B1\_wiener2=wiener2(B\_1,[5,5]);

%第二次维纳滤波

H = fspecial('gaussian',[5 5]);

R1\_Blurred2 = imfilter(R\_1,H);

G1\_Blurred2 = imfilter(G\_1,H);

B1\_Blurred2 = imfilter(B\_1,H);

%高斯滤波

RGB\_junzhi(:,:,1)=R1\_junzhi(:,:,1);

RGB\_junzhi(:,:,2)=G1\_junzhi(:,:,1);

RGB\_junzhi(:,:,3)=B1\_junzhi(:,:,1);

RGB\_zhongzhi(:,:,1)=R1\_zhongzhi(:,:,1);

RGB\_zhongzhi(:,:,2)=G1\_zhongzhi(:,:,1);

RGB\_zhongzhi(:,:,3)=B1\_zhongzhi(:,:,1);

RGB\_wiener1(:,:,1)=R1\_wiener1(:,:,1);

RGB\_wiener1(:,:,2)=G1\_wiener1(:,:,1);

RGB\_wiener1(:,:,3)=B1\_wiener1(:,:,1);

RGB\_wiener2(:,:,1)=R1\_wiener2(:,:,1);

RGB\_wiener2(:,:,2)=G1\_wiener2(:,:,1);

RGB\_wiener2(:,:,3)=B1\_wiener2(:,:,1);

RGB\_Blurred2(:,:,1)=R1\_Blurred2(:,:,1);

RGB\_Blurred2(:,:,2)=G1\_Blurred2(:,:,1);

RGB\_Blurred2(:,:,3)=B1\_Blurred2(:,:,1);

figure(2);

subplot(2,3,1);imshow(I);title('原图像');

subplot(2,3,2);imshow(I2);title('加入乘性噪声后');

subplot(2,3,3);imshow(RGB\_junzhi);title('均值滤波后');

subplot(2,3,4);imshow(RGB\_zhongzhi);title('中值滤波后');

subplot(2,3,5);imshow(RGB\_wiener2);title('二次维纳滤波后');

subplot(2,3,6);imshow(RGB\_Blurred2);title('高斯滤波后');

%subplot(2,3,6);imshow(RGB\_wiener1);title('一次维纳滤波后');

R\_2 = I1(:,:,1);

G\_2 = I1(:,:,2);

B\_2 = I1(:,:,3);

h1=fspecial('average',3);

%fspecial用于产生预定义滤波器

R2\_junzhi=uint8(round(filter2(h1,R\_2)));

G2\_junzhi=uint8(round(filter2(h1,G\_2)));

B2\_junzhi=uint8(round(filter2(h1,B\_2)));

%均值滤波，uin8为变量类型，round为取整，h为滤波参数即均值，h后的变量为要处理的图像

R2\_zhongzhi=medfilt2(R\_2,[5,5]);

G2\_zhongzhi=medfilt2(G\_2,[5,5]);

B2\_zhongzhi=medfilt2(B\_2,[5,5]);

%中值滤波

R2\_wiener1=wiener2(R\_2,[5,5]);

G2\_wiener1=wiener2(G\_2,[5,5]);

B2\_wiener1=wiener2(B\_2,[5,5]);

%第一次维纳滤波

R2\_wiener2=wiener2(R\_2,[5,5]);

G2\_wiener2=wiener2(G\_2,[5,5]);

B2\_wiener2=wiener2(B\_2,[5,5]);

%第二次维纳滤波

H1 = fspecial('gaussian',[5 5]);

R2\_Blurred2 = imfilter(R\_2,H1);

G2\_Blurred2 = imfilter(G\_2,H);

B2\_Blurred2 = imfilter(B\_2,H);

%高斯滤波

RGB1\_junzhi(:,:,1)=R2\_junzhi(:,:,1);

RGB1\_junzhi(:,:,2)=G2\_junzhi(:,:,1);

RGB1\_junzhi(:,:,3)=B2\_junzhi(:,:,1);

RGB1\_zhongzhi(:,:,1)=R2\_zhongzhi(:,:,1);

RGB1\_zhongzhi(:,:,2)=G2\_zhongzhi(:,:,1);

RGB1\_zhongzhi(:,:,3)=B2\_zhongzhi(:,:,1);

RGB1\_wiener1(:,:,1)=R2\_wiener1(:,:,1);

RGB1\_wiener1(:,:,2)=G2\_wiener1(:,:,1);

RGB1\_wiener1(:,:,3)=B2\_wiener1(:,:,1);

RGB1\_wiener2(:,:,1)=R2\_wiener2(:,:,1);

RGB1\_wiener2(:,:,2)=G2\_wiener2(:,:,1);

RGB1\_wiener2(:,:,3)=B2\_wiener2(:,:,1);

RGB1\_Blurred2(:,:,1)=R2\_Blurred2(:,:,1);

RGB1\_Blurred2(:,:,2)=G2\_Blurred2(:,:,1);

RGB1\_Blurred2(:,:,3)=B2\_Blurred2(:,:,1);

figure(3);

subplot(2,3,1);imshow(I);title('原图像');

subplot(2,3,2);imshow(I1);title('加入高斯噪声后');

subplot(2,3,3);imshow(RGB1\_junzhi);title('均值滤波后');

subplot(2,3,4);imshow(RGB1\_zhongzhi);title('中值滤波后');

subplot(2,3,5);imshow(RGB1\_wiener2);title('二次维纳滤波后');

subplot(2,3,6);imshow(RGB1\_Blurred2);title('高斯滤波后');

%subplot(2,3,6);imshow(RGB1\_wiener1);title('一次维纳滤波后');