哈尔滨工业大学（威海）

《数字图像处理》

实验报告

院系：计算机科学与技术学院

班级：20004401

学号：2200400327

姓名：王硕

实验 1	实验 2	总分

实验一

题目 1：基本 MATLAB 图像处理程序练习

1、基本 MATLAB 图像处理函数的使用

（1） imread 函数用来实现图像文件的读取。

```
A=imread('drum.bmp'); %用imread 函数来读入图像
imshow(A); %用 imshow 函数来显示图像

A=imread('C:\Users\Xin\Desktop\1.png');
imshow(A);
```

实验内容：尝试使用自己喜欢的图像练习该函数。

（2）imwrite 函数用来实现图像文件的写入。

```
imwrite(A,'C:\Users\Xin\Desktop\1.png');
```

实验内容：尝试使用自己喜欢的图像练习该函数。

（3）imfinfo 函数用来查询图像文件信息。

```
info=imfinfo('C:\Users\Xin\Desktop\1.png');%单步调试看info中成员
w=info.Width;%等其他信息
```

实验内容：尝试使用自己喜欢的图像练习该函数。

（4）subimage 函数实现在一个图形窗口中显示多幅图像。

```
RGB=imread('drum.bmp');
I=rgb2gray(RGB);
subplot(1,2,1),subimage(RGB) %subimage 实现在一个图形窗口中显示多幅图像
subplot(1,2,2),subimage(I)
```

实验内容：尝试使用自己喜欢的图像练习该函数。

2、图像的空间域操作

（1）imresize 函数实现图像的缩放。

```
J=imread('drum.bmp');
X1=imresize(J,2); %对图像进行缩放
figure,imshow(J)
```

实验内容：

1）单步调试，查看 J 和 X1 的大小，学习参数 2 的含义

2）使用 help 命令，查看 imresize 函数功能，将图像 J 转换为 20*50 大小图像并显示。

（2）imrotate 函数实现图像的旋转。

```
I=imread('drum.bmp');
J=imrotate(I,45,'bilinear'); %对图像进行旋转
```

```
subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(J);
```
实验内容：尝试使用自己喜欢的图像练习该函数。

（3）imcrop 函数实现图像的剪切。
```
I=imread('drum.bmp');
I2=imcrop(I,[75 68 130 112]); %对图像进行剪切
subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(I2);
```
实验内容：尝试使用自己喜欢的图像练习该函数。

（4）roicolor 函数是对 RGB 图像和灰度图像实现按灰度或亮度值选择区域进行处理。
```
a=imread('drum.bmp');
I=rgb2gray(a);
BW=roicolor(I,128,225); %按灰度值选择的区域
subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(BW)
```
实验内容：尝试使用自己喜欢的图像练习该函数，尝试不同灰度值的选取结果。

3、图像代数运算

（1）imadd 函数实现两幅图像的相加或者给一幅图像加上一个常数。给图像每个像素都增加亮度的程序如下：
```
I=imread('drum.bmp');
J=imadd(I,100); %给图像增加亮度
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```
实验内容：尝试使用自己喜欢的图像练习该函数，比较增加不同亮度的结果，实现两幅图像相加。

（2）imsubtract 函数实现将一幅图像从另一幅图像中减去，或者从一幅图像中减去一个常数。
```
I=imread('drum.bmp');
J=imsubtract(I,100); %给图像减少亮度
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```
实验内容：尝试使用自己喜欢的图像练习该函数，实现两幅图像相减。

（3）immultiply 实现两幅图像的相乘或者一幅图像的亮度缩放。
```
I=imread('drum.bmp');
J=immultiply(I,0.5); %对图像进行亮度缩放
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```
实验内容：尝试使用自己喜欢的图像练习该函数，实现两幅图像相乘。

（4）imdivide 函数实现两幅图像的除法或一幅图像的亮度缩放。

```
I=imread('drum.bmp');
J=imdivide(I,0.5); %图像的亮度缩放
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```

实验内容：尝试使用自己喜欢的图像练习该函数，实现两幅图像相除。

4、图像增强、变换和分割

（1）imhist 函数产生图像的直方图。

```
A=imread('drum.bmp'); %读入图像
B=rgb2gray(A); %把RGB 图像转化成灰度图像
imshow(B); %显示灰度图像
imhist(B); %显示灰度图像的直方图
```

（2）histeq 函数用于对图像的直方图均衡化。

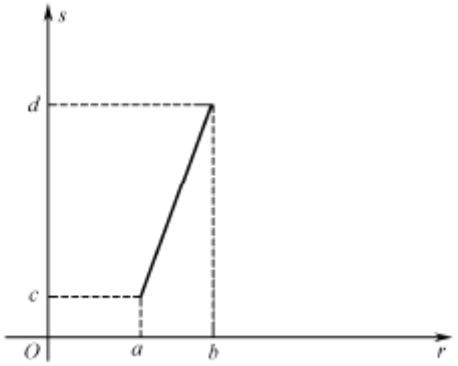
接上面程序输入以下程序：

```
C=histeq(B); %对图像B 进行均衡化
imshow(C); %显示图像
imhist(C); %得到均衡化后的灰度直方图
```

实验内容：尝试使用自己喜欢的图像练习该函数，比较处理前后的两幅图像效果与对应的直方图的变化。

（3）线性与非线性灰度映射。

线性映射函数s=T(r)：



非线性映射函数s=T(r)：

$$s = \mathrm{c}\log(1+r)$$

灰度倒置s=T(r)：

$$\mathrm{s} = 255\text{-r}$$

二值化：确定一阈值$r1$，$0<r1<255$。$r>r1$ 的灰度值置白，$r<r1$ 的灰度值置黑。

参考程序如下。

```
I=imread('i_lena.jpg');
figure;
subplot(1,3,1);
```

```
imshow(I);
title('原图');
J=imadjust(I,[0.3;0.6],[0.1;0.9]);
%设置输入/输出变换的灰度级范围，a=0.3, b=0.6, c=0.1, d=0.9。
subplot(1,3,2);
imshow(J);
title('线性映射');
I1=double(I);
I2=I1/255;
C=2;% 设置非线性映射函数的参数c=2。
K=C*log(1+I2); %求图像的对数变换
subplot(1,3,3);
imshow(K);
title('非线性映射');
M=255-I; %将此图像取反
figure;
subplot(1,3,1);
imshow(M);
title('灰度倒置');
N1=im2bw(I,0.4); %将此图像二值化，阈值为0.4
N2=im2bw(I,0.7); %将此图像二值化，阈值为0.7
subplot(1,3,2);
imshow(N1);
title('二值化阈值0.4');
subplot(1,3,3);
imshow(N2);
title('二值化阈值0.7');
```

实验内容：

1）分析调试上面代码，查看其中各步骤变量类型与值的变化，分析程序功能。

2）分别查看*d-c*大于和小于*b-a*的情况，观察输出图像的对比度增大减小情况。

3）分析查看对数非线性映射使图像的高、低灰度级的变化情况。

4）设置不同参数进行实验，思考线性映射与对数非线性映射的图像处理效果的区别。

4）查看灰度倒置的图像效果。

5）选择不同的阈值，观察阈值对图像二值化的影响。

（4）filter2 函数实现均值滤波。

```
a=imread('noise.drum.jpg');
I=rgb2gray(a);
imshow(I);
K1=filter2(fspecial('average',3),I)/255; %3×3 的均值滤波
K2=filter2(fspecial('average',5),I)/255; %5×5 的均值滤波
K3=filter2(fspecial('average',7),I)/255; %7×7 的均值滤波
figure,imshow(K1);
figure,imshow(K2);
```

```
figure,imshow(K3);
```
（5）medfilt2 函数实现中值滤波。
```
a=imread('noise.drum.jpg');
I=rgb2gray(a);
imshow(I);
K1=medfilt2(I,[3,3]); %3×3 中值滤波
K2=medfilt2(I,[5,5]); %5×5 中值滤波
K3=medfilt2(I,[7,7]); %7×7 中值滤波
figure,imshow(K1);
figure,imshow(K2);
figure,imshow(K3);
```
实验内容：
1）增大滤波器模板，观察图像效果变化，比较均值滤波和中值滤波的效果差别。
2）加入高斯噪声（P1=imnoise(M,'gaussian',0.02);）和椒盐噪声（P2=imnoise(M,'salt & pepper',0.09);）观察不同模板尺寸下，均值滤波和中值滤波对两种噪声的滤波效果差别。

（6）fft2 函数和ifft2 函数分别是计算二维的傅里叶变换和反变换。
```
f=zeros(100,100);
f(20:70,40:60)=1;
imshow(f);
F=fft2(f);
F2=log(abs(F)); %对幅值取对数
imshow(F2),colorbar
```
实验内容：分析调试上面代码，查看其中各步骤变量类型与值的变化，分析程序功能。

（7）fftshift 函数实现了补零操作和改变图像显示象限。
```
f=zeros(100,100);
f(20:70,40:60)=1;
imshow(f);
F=fft2(f,256,256);
F2=fftshift(F); %实现补零操作
imshow(log(abs(F2)));
```
实验内容：分析调试上面代码，查看其中各步骤变量类型与值的变化，分析程序功能。

（8）edge 函数用于提取图像的边缘。
输入以下程序：
```
I=imread('cameraman.tif');
%I=rgb2gray(I);
BW=edge(I,'prewitt');
imshow(I),figure,imshow(BW,[]);
```
实验内容：比较canny，prewitt，sobel，roberts，log算子的边缘检测效果。

题目1的运行代码、使用参数、实验结果截图、实验结论：

* 1.1-1.3较为基础，直接展示代码：

```
%1.1.1 imread 函数用来实现图像文件的读取A=imread('1.png');
imshow(A);

%1.1.2 imwrite 函数用来实现图像文件的写入imwrite(A,'2.png');

%1.1.3 imfinfo 函数用来查询图像文件信息info=imfinfo('1.png');
w=info.Width;
```

* 1.4 实现在一个图形窗口中显示多幅图像，代码如下，运行截图为图1

```
%1.1.4 subimage 函数实现在一个图形窗口中显示多幅图像
RGB=imread('2.png');
I=rgb2gray(RGB);
subplot(1,2,1),subimage(RGB)
subplot(1,2,2),subimage(I)
```

* 2.1 函数实现图像的缩放，单步调试，查看 J 和 X1 的大小即可发现参数“2”是放大2倍的意思，更改参数为二元组[a b]还可将图片改为a*b尺寸的图片，代码如下，运行截图为图2

```
% 1.2.1 imresize 函数实现图像的缩放
J=imread('drum.bmp');
X1=imresize(J,2); %2是2倍的意思
figure,imshow(J)

J=imread('1.png');
X1=imresize(J,[20 50]);
figure,imshow(X1)
```

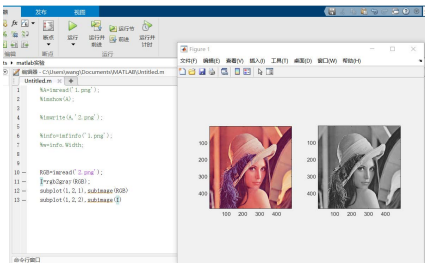


图1

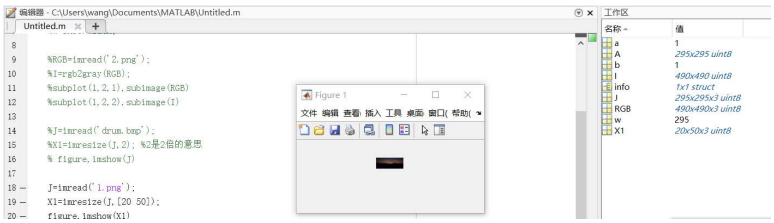


图2

* 2.2函数实现图像的旋转，参数90代表旋转90°，代码如下，运行截图为图3

```
%1.2.2 imrotate 函数实现图像的旋转
I=imread('1.png');
J=imrotate(I,90,'bilinear');
subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(J);
```

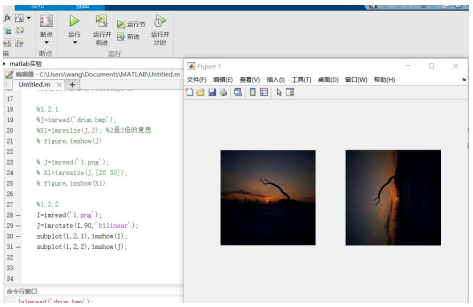


图3

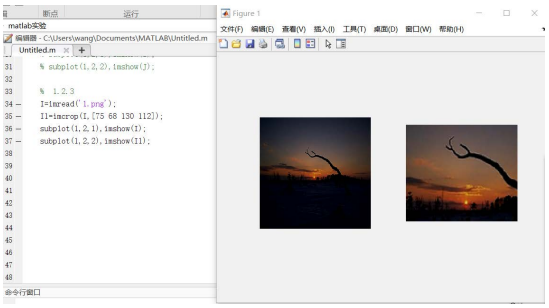


图4

* 2.3 函数实现图像的剪切，imcrop函数的参数四元组代表从某行到某行和某列到某列的剪切，代码如下，运行截图为图4

```
%1.2.3 imcrop 函数实现图像的剪切
I=imread('1.png');
I1=imcrop(I,[75 68 130 112]);
subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(I1);
```

* 2.4 函数对 RGB 图像和灰度图像实现按灰度或亮度值选择区域进行处理，函数参数为灰度范围，代码如下，运行截图为图5

```
% 1.2.4 roicolor对 RGB 图像和灰度图像实现按灰度或亮度值选择区域进行处理
A=imread('2.png');
I=rgb2gray(A);
BW=roicolor(I,15,150);
subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(BW)
```

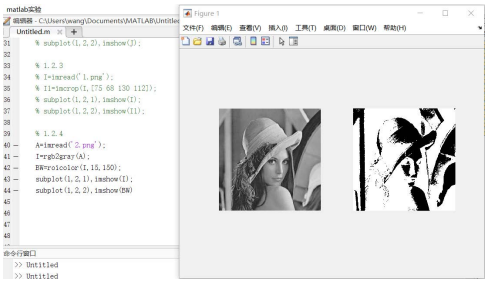


图5

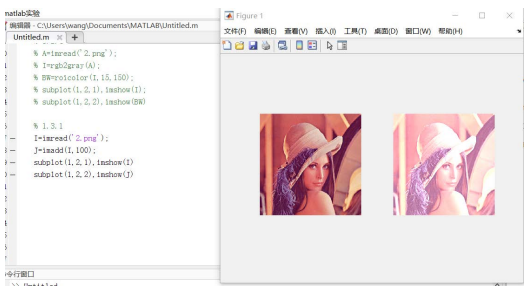


图6

* 3.1 函数实现两幅图像的相加或者给一幅图像加上一个常数，代码如下，运行截图为图6

```
%1.3.1 imadd 函数实现两幅图像的相加或者给一幅图像加上一个常数。给图像每个像素都增加亮度
I=imread('2.png');
J=imadd(I,100);
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```

* 3.2 函数实现将一幅图像从另一幅图像中减去，或者从一幅图像中减去一个常数，代码如下，运行截图为图7

```
%1.3.2 imsubtract 函数实现将一幅图像从另一幅图像中减去，或者从一幅图像中减去一个常数
I=imread('2.png');
J=imsubtract(I,150);
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```

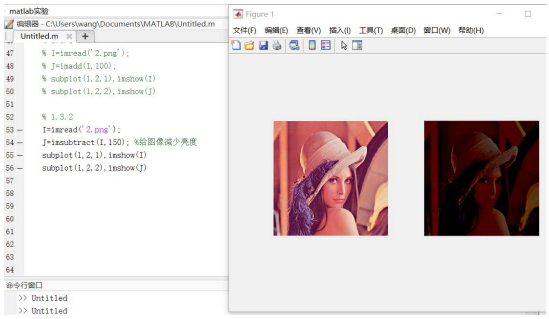


图7

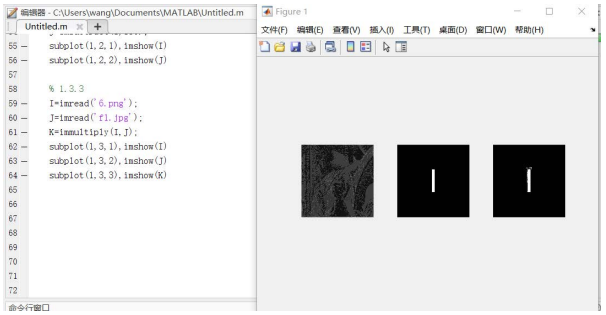


图8

* 3.3 函数实现两幅图像的相乘或者一幅图像的亮度缩放，代码如下，运行截图为图8

```
%1.3.3 immultiply 实现两幅图像的相乘或者一幅图像的亮度缩放
I=imread('6.png');
J=imread('f1.jpg');
K=immultiply(I,J);
subplot(1,3,1),imshow(I)
subplot(1,3,2),imshow(J)
subplot(1,3,3),imshow(K)
```

* 3.4 函数实现两幅图像的除法或一幅图像的亮度缩放，参数0.5为缩放倍数，代码如下，运行截图为图9

```
%1.3.4 imdivide 函数实现两幅图像的除法或一幅图像的亮度缩放
I=imread('2.png');
J=imdivide(I,0.5); %图像的亮度缩放
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```

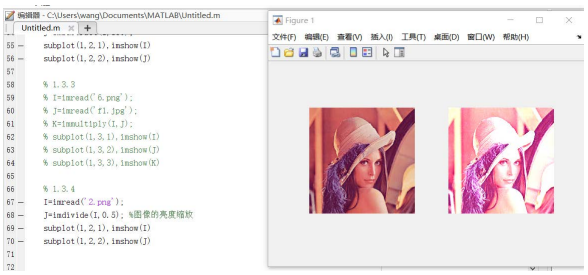


图9

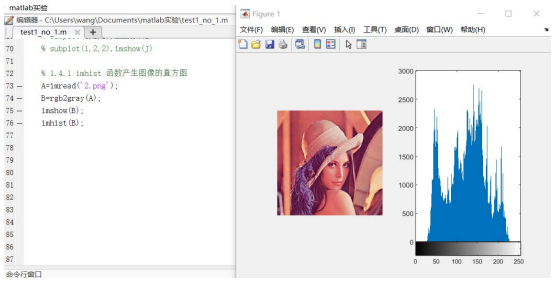


图10

* 4.1 函数函数产生图像的直方图，代码如下，运行截图为图10

```
%1.4.1 imhist 函数产生图像的直方图
A=imread('2.png');
B=rgb2gray(A);
imshow(B);
imhist(B);
```

* 4.2 函数用于对图像的直方图均衡化，代码如下，运行截图为图11

```
% 1.4.2 histeq 函数用于对图像的直方图均衡化
A=imread('2.png');
B=rgb2gray(A);
imshow(B);
imhist(B);
C=histeq(B);
imshow(C);
imhist(C);
```

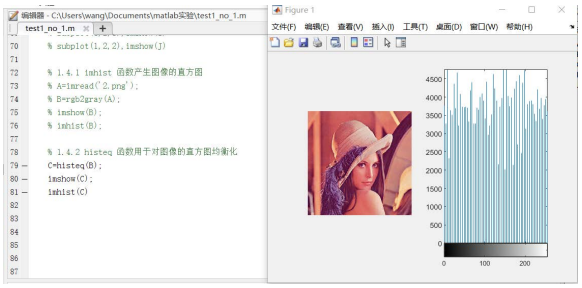


图11

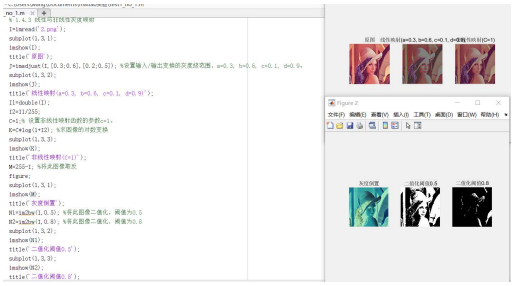


图12

* 4.3 线性与非线性灰度映射，代码如下，运行截图为图12

```
%1.4.3 线性与非线性灰度映射
I=imread('2.png');
subplot(1,3,1);
imshow(I);
title('原图');
J=imadjust(I,[0.3;0.6],[0.2;0.5]); %设置输入/输出变换的灰度级范围，a=0.3,b=0.6,
c=0.1, d=0.9
subplot(1,3,2);
imshow(J);
title('线性映射(a=0.3, b=0.6, c=0.1, d=0.9)');
I1=double(I);
I2=I1/255;
C=1;% 设置非线性映射函数的参数c=1。
K=C*log(1+I2); %求图像的对数变换
subplot(1,3,3);
imshow(K);
title('非线性映射(C=1)');
```

```
M=255-I; %将此图像取反
figure;
subplot(1,3,1);
imshow(M);
title('灰度倒置');
N1=im2bw(I,0.5); %将此图像二值化，阈值为0.5
N2=im2bw(I,0.8); %将此图像二值化，阈值为0.8
subplot(1,3,2);
imshow(N1);
title('二值化阈值0.5');
subplot(1,3,3);
imshow(N2);
title('二值化阈值0.8');
```

从实验4.3中可以看出：

1. 当线性映射的d-c<b-a时，图像对比度增大，反之则减小。
2. 对数非线性映射中，高灰度区域对比度下降，低灰度区域对比度增加，操作对低灰度级的效果不明显，对高灰度级效果明显。
3. 线性映射直接均衡地改变全局对比度和亮度，非线性映射则可以对特定灰度区域像素进行处理。
4. 合理选择二值化阈值可实现基本的图像分割。

* 4.4 函数实现均值滤波，代码如下，运行截图为图13

```
%1.4.4 函数实现均值滤波。
a=imread('2.png');
I=rgb2gray(a);
subplot(2,2,1),imshow(I);
title('灰度图');
K1=filter2(fspecial('average',3),I)/255; %3×3 的均值滤波
K2=filter2(fspecial('average',5),I)/255; %5×5 的均值滤波
K3=filter2(fspecial('average',7),I)/255; %7×7 的均值滤波
subplot(2,2,2),imshow(K1);
title('3×3均值滤波');
subplot(2,2,3),imshow(K2);
title('5×5均值滤波');
subplot(2,2,4),imshow(K3);
title('7×7均值滤波');
```

* 4.5 函数实现中值滤波，代码如下，运行截图为图14

```
% 1.4.5 medfilt2 函数实现中值滤波。
a=imread('2.png');
I=rgb2gray(a);
subplot(2,2,1),imshow(I);
title('灰度图');
K1=medfilt2(I,[3,3]); %3×3 中值滤波
K2=medfilt2(I,[5,5]); %5×5 中值滤波
K3=medfilt2(I,[7,7]); %7×7 中值滤波
```

```
subplot(2,2,2),imshow(K1);
title('3×3中值滤波');
subplot(2,2,3),imshow(K2);
title('5×5中值滤波');
```

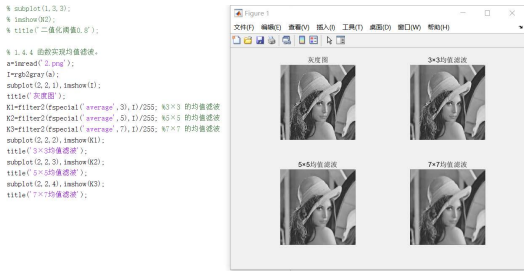


图13

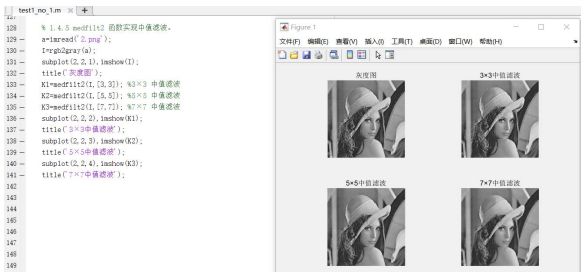


图14

* 去噪效果对比

加入高斯噪声和椒盐噪声，不同模板尺寸下，均值滤波和中值滤波特点均为模板越大，去噪效果越好，但同时处理后的图像也越模糊；无噪声时两种模板滤波效果差别不大，对两种噪声的滤波效果差别明显，均值滤波对高斯噪声的效果明显比中值滤波好，中值滤波对椒盐噪声的效果明显比均值滤波好，代码如下对比图见图15

```
a=imread('2.png');
I=rgb2gray(a);
P1=imnoise(I,'gaussian',0.02);
subplot(2,3,1),imshow(P1);
title('高斯噪声');
P2=imnoise(I,'salt & pepper',0.09);
subplot(2,3,4),imshow(P2);
title('椒盐噪声');
K1=filter2(fspecial('average',3),P1)/255; %3×3 的均值滤波
subplot(2,3,2),imshow(K1);
title('3×3均值滤波');
K2=medfilt2(P1,[3,3]); %3×3 中值滤波
subplot(2,3,3),imshow(K2);
title('3×3中值滤波');
K3=filter2(fspecial('average',3),P2)/255; %3×3 的均值滤波
subplot(2,3,5),imshow(K3);
title('3×3均值滤波');
K4=medfilt2(P2,[3,3]); %3×3 中值滤波
subplot(2,3,6),imshow(K4);
title('3×3中值滤波');
```

```
%去噪效果对比
a=imread('2.png');
% I=rgb2gray(a);
P1=imnoise(I,'gaussian',0.02);
subplot(2,3,1),imshow(P1);
title('高斯噪声');
P2=imnoise(I,'salt & pepper',0.09);
subplot(2,3,4),imshow(P2);
title('椒盐噪声');
K1=filter2(fspecial('average',3),P1)/255; %3×3 的均值滤波
subplot(2,3,2),imshow(K1);
title('3×3均值滤波');
K2=medfilt2(P1,[3,3]); %3×3 中值滤波
subplot(2,3,3),imshow(K2);
title('3×3中值滤波');
K3=filter2(fspecial('average',3),P2)/255; %3×3 的均值滤波
subplot(2,3,5),imshow(K3);
title('3×3均值滤波');
K4=medfilt2(P2,[3,3]); %3×3 中值滤波
subplot(2,3,6),imshow(K4);
title('3×3中值滤波');
```

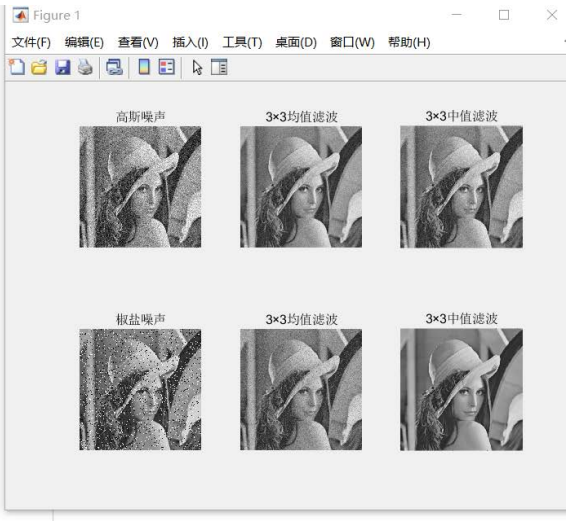


图15

* 4.6 函数计算二维的傅里叶变换和反变换，代码如下，运行截图为图16

```
%1.4.6 fft2 函数和ifft2 函数计算二维的傅里叶变换和反变换
f=zeros(100,100);
f(30:70,40:60)=1;
subplot(2,2,1),imshow(f);
title('f');
F=fft2(f);
subplot(2,2,2),imshow(F);
title('F');
F1=ifft2(F);
subplot(2,2,3),imshow(F1);
title('F1');
F2=log(abs(F)); %对幅值取对数
subplot(2,2,4),imshow(F2),colorbar;
title('F2');
```

* 4.7 函数实现补零操作和改变图像显示象限，代码如下，运行截图为图14

```
% 1.4.7 fftshift函数实现补零操作和改变图像显示象限
f=zeros(100,100);
f(20:70,40:60)=1;
imshow(f);
F=fft2(f,256,256);
F2=fftshift(F); %实现补零操作
imshow(log(abs(F2)));
```

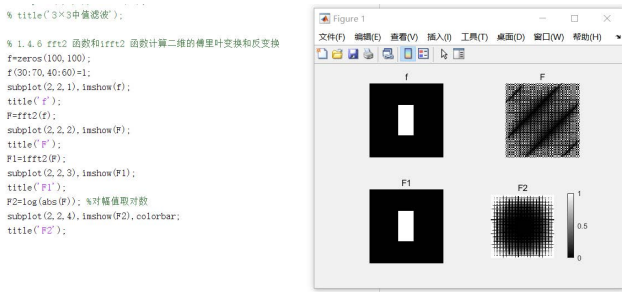


图16

图17

4.8函数计算二维的傅里叶变换和反变换，代码如下，运行截图为图18

```
%1.4.8 edge 函数用于提取图像的边缘。
I=imread('cameraman.tif');
BW1=edge(I,'canny');
BW2=edge(I,'prewitt');
BW3=edge(I,'sobel');
BW4=edge(I,'roberts');
BW5=edge(I,'log');
subplot(2,3,1),imshow(I),title('I');
subplot(2,3,2),imshow(BW1),title('canny');
subplot(2,3,3),imshow(BW2),title('prewitt');
subplot(2,3,4),imshow(BW3),title('sobel');
subplot(2,3,5),imshow(BW4),title('roberts');
subplot(2,3,6),imshow(BW5),title('log')
```

```
% F=fft2(I,256,256);
% F2=fftshift(F); %实现补零操作
% imshow(log(abs(F2)));

% 1.4.8 edge 函数用于提取图像的边缘。
I=imread('cameraman.tif');
BW1=edge(I,'canny');
BW2=edge(I,'prewitt');
BW3=edge(I,'sobel');
BW4=edge(I,'roberts');
BW5=edge(I,'log');
subplot(2,3,1),imshow(I),title('I');
subplot(2,3,2),imshow(BW1),title('canny');
subplot(2,3,3),imshow(BW2),title('prewitt');
subplot(2,3,4),imshow(BW3),title('sobel');
subplot(2,3,5),imshow(BW4),title('roberts');
subplot(2,3,6),imshow(BW5),title('log');
```

图18

题目2、深入MATLAB图像处理程序语句练习

1、图像采样

```
a = imread('e:\i_lena.JPG');
b = rgb2gray(a);
[wid,hei]=size(b);
%4 倍减采样
quartimg = zeros(wid/2+1,hei/2+1);
i1 = 1;
j1 = 1;
for i=1:2:wid
    for j=1:2:hei
        quartimg(i1,j1) = b(i,j);
        j1 = j1 + 1;
    end
    i1 = i1 + 1;
    j1 = 1;
end
figure
imshow(uint8(quartimg))
```

实验内容：

1）运行并分析以上代码，学习matlab中for语句的使用方法，并逐行解释上述代码。

2）尝试对图像的16倍减采样，在同一窗口中查看其原图、4倍、16倍减采样效果。

3）思考将一幅图如果进行4 倍、16 倍和64 倍增采样会出现什么情况？

（有可能出现错误：wid/2+1,hei/2+1不是整数，这是需要将原图像resize成偶数长宽）

2、直方图规定化处理（选做）

```
I=imread('C:\Users\Xin\Desktop\f1.jpg');
clear all
close all
%
I=double(imread('cameraman.tif'));
figure,imshow(I,[])
N=32;
Hist_image=hist(I(:),N);
Hist_image=Hist_image/sum(Hist_image);
Hist_image_cumulation=cumsum(Hist_image);
figure,stem([0:N-1],Hist_image);
%1.
Index=0:N-1;
%
Hist{1}=exp(-(Index-4).^2/8);
```

```
Hist{1}=Hist{1}/sum(Hist{1});
Hist_cumulation{1}=cumsum(Hist{1});
figure,stem([0:N-1],Hist{1})
%
Hist{2}=abs(2*N-1-2*Index);
Hist{2}=Hist{2}/sum(Hist{2});
Hist_cumulation{2}=cumsum(Hist{2});
figure,stem([0:N-1],Hist{2})
%2.
for m=1:2 %
    Image=I;
    %2.1
    for k=1:N
        Temp=abs(Hist_image_cumulation(k)-Hist_cumulation{m});
        [Temp1,Project{m}(k)]=min(Temp);
    end
    %
    for k=1:N
        Temp=find(Project{m}==k);
        if isempty(Temp)
            Hist_result{m}(k)=0;
        else
            Hist_result{m}(k)=sum(Hist_image(Temp));
        end
    end
    figure,stem([0:N-1],Hist_result{m});
    %
    Step=256/N;
    for k=1:N
    Index=find(I>=Step*(k-1)&I<Step*k);
    Image(Index)=Project{m}(k);
    end
    figure,imshow(Image,[])
end
```

实验内容：

1）逐行调试代码分析查看变量，熟悉matlab向量操作，对程序逐行给出注解。

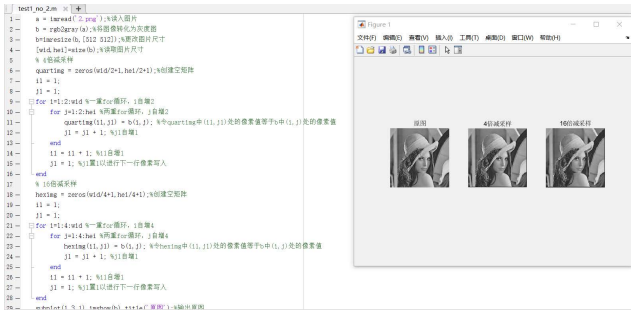
2）分析图像处理效果，给出结论。

3）尝试自己设计目标直方图，将输入图像按目标直方图进行规定化处理。

题目2的运行代码、使用参数、实验结果截图、实验结论：

图像采样

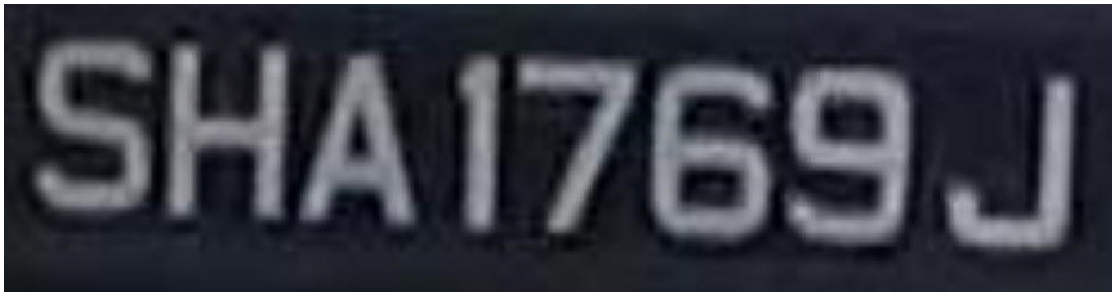
for循环实现图像采样，两重for循环，步长分别为2和4以实现4倍和16倍减采样，代码如下，运行截图见下图



```
a = imread('2.png');%读入图片
b = rgb2gray(a);%将图像转化为灰度图
b=imresize(b,[512 512]);%更改图片尺寸
[wid,hei]=size(b);%读取图片尺寸
% 4倍减采样
quartimg = zeros(wid/2+1,hei/2+1);%创建空矩阵
i1 = 1;
j1 = 1;
for i=1:2:wid %一重for循环，i自增2
   for j=1:2:hei %两重for循环，j自增2
      quartimg(i1,j1) = b(i,j); %令quartimg中(i1,j1)处的像素值等于b中(i,j)处的像素值
      j1 = j1 + 1; %j1自增1
   end
   i1 = i1 + 1; %i1自增1
   j1 = 1; %j1置1以进行下一行像素写入
end
% 16倍减采样
heximg = zeros(wid/4+1,hei/4+1);%创建空矩阵
i1 = 1;
j1 = 1;
for i=1:4:wid %一重for循环，i自增4
   for j=1:4:hei %两重for循环，j自增4
      heximg(i1,j1) = b(i,j); %令heximg中(i1,j1)处的像素值等于b中(i,j)处的像素值
      j1 = j1 + 1; %j1自增1
   end
   i1 = i1 + 1; %i1自增1
   j1 = 1; %j1置1以进行下一行像素写入
end
subplot(1,3,1),imshow(b),title('原图');%输出原图
subplot(1,3,2),imshow(uint8(quartimg)),title('4倍减采样');%输出4倍减采样图像
subplot(1,3,3),imshow(uint8(heximg)),title('16倍减采样');%输出4倍减采样图像
```

实验二

题目1、车牌分割及图像傅里叶变换性质验证



实验内容：

1）针对以上车牌图像，计算其图像直方图，选择合适的阈值进行二值化，而后基于二值图像编程分割出每个字符，将其缩放为统一标准大小100*150，而后放入256*256的黑色背景图像的中间位置。

2）查看生成的各个字符图像（256*256）的傅里叶频谱图。

3）选择其中一个字符图像（256*256），将其中的100*150字符平移一定位置，再次计算其傅里叶频谱；对平移后的字符图像（256*256）进行90度旋转，再次计算其频谱图；将图像中的字符放大或缩小一定量（仍放在256*256黑色背景中），再次计算其频谱图；将图像取反，再次计算其频谱图。在同一窗口中显示5张原图与5张频谱图，查看傅里叶变换的平移、旋转、翻转特性，总结频谱图高低频成分与图像空间梯度的关系。

题目1的运行代码、使用参数、实验结果截图、实验结论：

车牌分割及图像傅里叶变换性质验证

首先计算其图像直方图，如图1，随后选择25%处灰度值为阈值进行二值化，更改图像尺寸以便于下一步分割

```
I=imread('carplate.jpg');
I1=rgb2gray(I); %转为灰度图
% imshow(I1);
I2=im2bw(I1,0.25); %#ok<*IM2BW>二值化
% imshow(I2);
I3=imresize(I2,[256 1024]);%更改尺寸
% imshow(I3);
```

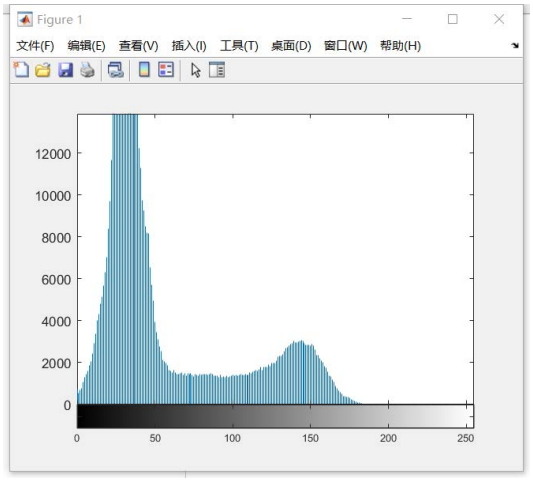


图1

```
%确定参数以将各数字切割并缩放
p1=imcrop(I2,[9 1 129 210]);
p1=imresize(p1,[100 150]);
p2=imcrop(I2,[130 1 127 220]);
p2=imresize(p2,[100 150]);
p3=imcrop(I2,[248 1 127 240]);
p3=imresize(p3,[100 150]);
p4=imcrop(I2,[380 10 60 220]);
p4=imresize(p4,[100 150]);
p5=imcrop(I2,[440 20 120 230]);
p5=imresize(p5,[100 150]);
p6=imcrop(I2,[550 20 128 256]);
p6=imresize(p6,[100 150]);
p7=imcrop(I2,[670 20 128 256]);
p7=imresize(p7,[100 150]);
p8=imcrop(I2,[810 20 128 256]);
p8=imresize(p8,[100 150]);
subplot(2,4,1),imshow(p1);
subplot(2,4,2),imshow(p2);
subplot(2,4,3),imshow(p3);
subplot(2,4,4),imshow(p4);
subplot(2,4,5),imshow(p5);
subplot(2,4,6),imshow(p6);
subplot(2,4,7),imshow(p7);
```

图2

基于二值图像编程分割出每个字符，将其缩放为统一标准大小100*150，分离出的字符，如图2。

```
% 确定参数以将各数字切割并缩放
p1=imcrop(I2,[9 1 129 210]);
p1=imresize(p1,[100 150]);
p2=imcrop(I2,[130 1 127 220]);
p2=imresize(p2,[100 150]);
p3=imcrop(I2,[248 1 127 240]);
p3=imresize(p3,[100 150]);
p4=imcrop(I2,[380 10 60 220]);
p4=imresize(p4,[100 150]);
p5=imcrop(I2,[440 20 120 230]);
p5=imresize(p5,[100 150]);
p6=imcrop(I2,[550 20 128 256]);
p6=imresize(p6,[100 150]);
p7=imcrop(I2,[670 20 128 256]);
p7=imresize(p7,[100 150]);
p8=imcrop(I2,[810 20 128 256]);
p8=imresize(p8,[100 150]);
```

将分离出的每个字符，放入256*256的黑色背景图像的中间位置，分理离出的字符，如图3。

```
% 放入黑色背景图中间
j1=zeros(256,256);
j2=zeros(256,256);
j3=zeros(256,256);
j4=zeros(256,256);
j5=zeros(256,256);
j6=zeros(256,256);
j7=zeros(256,256);
j8=zeros(256,256);
j1(79:178,54:203)=p1;
j2(79:178,54:203)=p2;
j3(79:178,54:203)=p3;
j4(79:178,54:203)=p4;
j5(79:178,54:203)=p5;
j6(79:178,54:203)=p6;
j7(79:178,54:203)=p7;
j8(79:178,54:203)=p8;
```

```
%放入黑色背景图中间
j1=zeros(256,256);
j2=zeros(256,256);
j3=zeros(256,256);
j4=zeros(256,256);
j5=zeros(256,256);
j6=zeros(256,256);
j7=zeros(256,256);
j8=zeros(256,256);
j1(70:169,54:203)=p1;
j2(70:169,54:203)=p2;
j3(70:169,54:203)=p3;
j4(70:169,54:203)=p4;
j5(70:169,54:203)=p5;
j6(70:169,54:203)=p6;
j7(70:169,54:203)=p7;
j8(70:169,54:203)=p8;
subplot(2,4,1),imshow(j1);
subplot(2,4,2),imshow(j2);
subplot(2,4,3),imshow(j3);
subplot(2,4,4),imshow(j4);
subplot(2,4,5),imshow(j5);
subplot(2,4,6),imshow(j6);
subplot(2,4,7),imshow(j7);
subplot(2,4,8),imshow(j8);
```

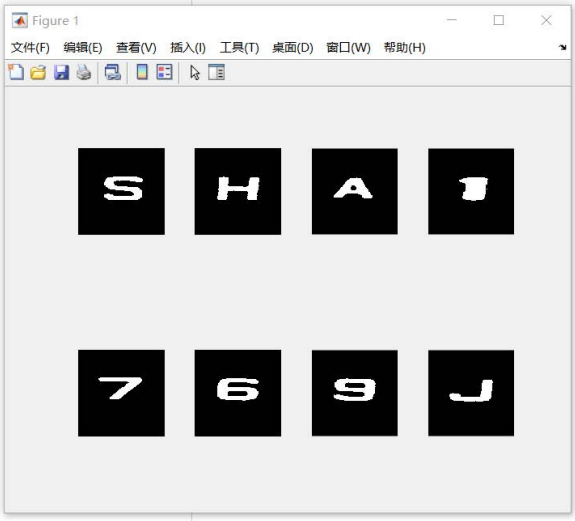


图3

查看生成的各个字符图像（256*256）的傅里叶频谱图，如图4。

```
% 傅里叶变换
f1=fft2(j1);
f1 = fftshift(f1);
f1 = abs(f1);
f2=fft2(j2);
f2 = fftshift(f2);
f2 = abs(f2);
f3=fft2(j3);
f3 = fftshift(f3);
f3 = abs(f3);
f4=fft2(j4);
f4 = fftshift(f4);
f4 = abs(f4);
f5=fft2(j5);
f5 = fftshift(f5);
f5 = abs(f5);
f6=fft2(j6);
f6 = fftshift(f6);
f6 = abs(f6);
f7=fft2(j7);
f7 = fftshift(f7);
f7 = abs(f7);
f8=fft2(j8);
f8 = fftshift(f8);
f8 = abs(f8);
```

```
f2=fft2(j2);
f2 = fftshift(f2);
f2 = abs(f2);
f3=fft2(j3);
f3 = fftshift(f3);
f3 = abs(f3);
f4=fft2(j4);
f4 = fftshift(f4);
f4 = abs(f4);
f5=fft2(j5);
f5 = fftshift(f5);
f5 = abs(f5);
f6=fft2(j6);
f6 = fftshift(f6);
f6 = abs(f6);
f7=fft2(j7);
f7 = fftshift(f7);
f7 = abs(f7);
f8=fft2(j8);
f8 = fftshift(f8);
f8 = abs(f8);
subplot(2,4,1),imshow(uint8(f1));
subplot(2,4,2),imshow(uint8(f2));
subplot(2,4,3),imshow(uint8(f3));
subplot(2,4,4),imshow(uint8(f4));
subplot(2,4,5),imshow(uint8(f5));
subplot(2,4,6),imshow(uint8(f6));
subplot(2,4,7),imshow(uint8(f7));
```

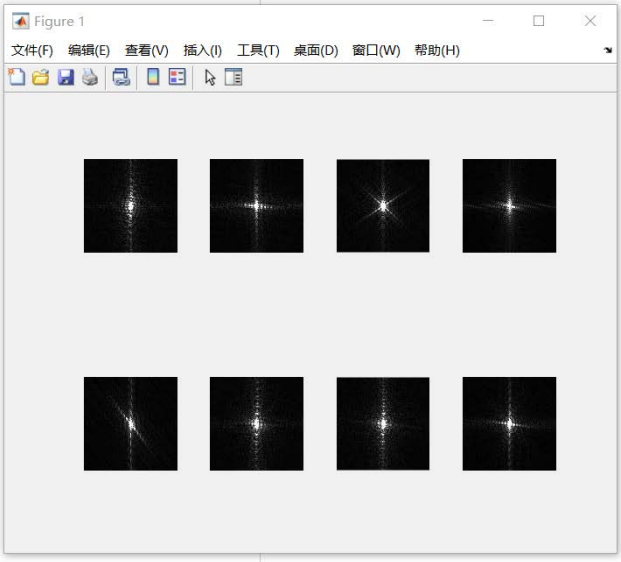


图4

选择第一个字符图像，将其中的100*150字符向右平移40像素，再次计算其傅里叶频谱；对平移后的字符图像进行90度旋转，再次计算其频谱图；将图像中的字符放大到1.2倍，再次计算其频谱图；将图像取反，再次计算其频谱图。在同一窗口中显示5张原图与5张频谱图，如图5。

对于频谱图高低频成分与图像空间梯度的关系：图像中的高频成分为图像灰度值剧烈变化的地方，即就是图像中的边缘。低频成分为图像中灰度值变化缓慢的区域，即所谓的平坦区域。图像中的边缘点是梯度图像的局部极值点，而不是原始图像的局部极值点，在该点处的梯度值局部最大，梯度方向为arctan(Δy/Δx)，梯度方向的范围为（-90°，90°），边缘点处的梯度角的绝对值大，在进行图像处理时可根据梯度方向来判断边缘的方向，因而平移/旋转/放缩/取反中只有旋转会使频谱图发生变化。

```
% 平移/旋转/放缩/取反
 t1=zeros(256,256);%平移
 t1(100:199,104:253)=p1;
 t2=imrotate(t1,90,'bilinear');%旋转
 tp=imresize(p1,1.2);%放缩
 t3=zeros(256,256);
 t3(69:188,39:218)=tp;
 t4=1-j1;%取反
 ft1=fft2(t1);
 ft1 = fftshift(ft1);
 ft1 = abs(ft1);
 ft2=fft2(t2);
 ft2 = fftshift(ft2);
 ft2 = abs(ft2);
 ft3=fft2(t3);
 ft3 = fftshift(ft3);
 ft3 = abs(ft3);
 ft4=fft2(t4);
 ft4 = fftshift(ft4);
 ft4 = abs(ft4);
 subplot(2,5,1),imshow(j1),title('原图');
 subplot(2,5,2),imshow(t1),title('平移');
 subplot(2,5,3),imshow(t2),title('旋转');
 subplot(2,5,4),imshow(t3),title('放缩');
 subplot(2,5,5),imshow(t4),title('取反');
 subplot(2,5,6),imshow(uint8(f1)),title('原图频谱图');
 subplot(2,5,7),imshow(uint8(ft1)),title('平移频谱图');
 subplot(2,5,8),imshow(uint8(ft2)),title('旋转频谱图');
 subplot(2,5,9),imshow(uint8(ft3)),title('放缩频谱图');
 subplot(2,5,10),imshow(uint8(ft4)),title('取反频谱图');
```

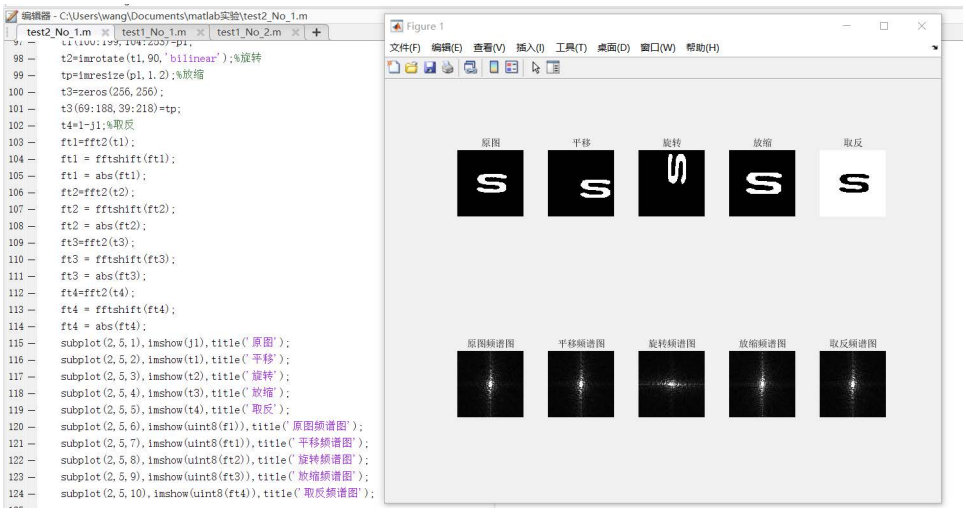


图5

题目2、图像的离散余弦变换编解码

dct2 函数实现离散余弦变换。与之对应，idct2 函数实现图像的二维逆离散余弦变换。

```
I = imread('cameraman.tif');
figure(1)
imshow(I)
%I = rgb2gray(RGB);
J = dct2(I);
figure(2)
imshow(log(abs(J)),[])
figure(3);
J(abs(J) < 10) = 0;
K = idct2(J)/255;
imshow(K)
```

实验内容：

1）逐行分析以上离散余弦变换程序，查看各步骤变量类型与值的变化，分析程序功能。

2）总结余弦变换系数的能量集中特性。分别选取不同的高、低频丢弃范围，观测恢复图像的质量变化。

3）（选做）对一副图像进行8*8分块dct变换编码。即每个8*8块进行dct变换，而后8*8变换系数与模板

```
mask1= [ 1 1 1 1 0 0 0 0
1 1 1 0 0 0 0 0
1 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0];
```

相乘进行高频截断（两个矩阵对应元素相乘用T=T1.*T2;），而后进行逆变换，最后把所有8*8块拼在一起，成为解码恢复图像。观察图像的原始dct系数与分块截断后的dct系数矩阵；比较原图像与恢复图像的效果差别；采用不同的截断矩阵（分别截断高低频），查看图像效果变化。

题目2的运行代码、使用参数、实验结果截图、实验结论：

图像的离散余弦变换编解码

dct2 函数实现离散余弦变换，与之对应，idct2 函数实现图像的二维逆离散余弦变换，代码和运行截图如下

余弦变换系数的能量集中特性：大多数的自然信号(包括声音和图像)的能量都集中在离散余弦变换后的低频部分故丢低频后图像损失较大

```
I = imread('5.png');%读入图像
I = rgb2gray(I);%转换为灰度图

J = dct2(I);%离散余弦变换
J1=J,J2=J;
J1(abs(J) >2000) = 0;%绝对值大于2000的置0，丢高频
J2(abs(J) < 50) = 0;%绝对值小于50的置0，丢低频

K = idct2(J)/255;%逆离散余弦变换还原
K1 = idct2(J1)/255
K2 = idct2(J2)/255

figure;imshow(I),title('原图');
figure;
subplot(2,3,1),imshow(log(abs(J)),[]),title('无丢弃');
subplot(2,3,2),imshow(log(abs(J1)),[]),title('丢高频');
subplot(2,3,3),imshow(log(abs(J2)),[]),title('丢低频');
subplot(2,3,4),imshow(K);
subplot(2,3,5),imshow(K1);
subplot(2,3,6),imshow(K2);
```