

**本科生实验报告**

**实验课程 数字图像处理技术**

**学院名称 核技术与自动化工程学院**

**专业名称 测控技术与仪器**

**学生姓名 追梦少年南南**

**学生学号 202006010336**

**指导教师 李晓丽**

**实验地点 6C702**

**实验成绩**

**二〇二二年三月 二〇二二年六月**

**填写说明**

1. 适用于本科生所有的实验报告（印制实验报告册除外）；
2. 专业填写为专业全称，有专业方向的用小括号标明；
3. 格式要求：
4. 用A4纸双面打印（封面双面打印）或在A4大小纸上用蓝黑色水笔书写。
5. 打印排版：正文用宋体小四号，1.5倍行距，页边距采取默认形式（上下2.54cm，左右2.54cm，页眉1.5cm，页脚1.75cm）。字符间距为默认值（缩放100%，间距：标准）；页码用小五号字底端居中。
6. 具体要求：

**题目**（二号黑体居中）；

正文部分采用三级标题；

**1.** ××(小二号黑体居中，段前0.5行)

**1.1** ×××××小三号黑体×××××（段前、段后0.5行）

**1.1.1**小四号黑体（段前、段后0.5行）

**实验一 MATLAB图像处理工具箱运用**

**一、实验目的**

通过实验掌握matlab图像处理工具箱的基本使用，灵活运用各种函数解决图像处理中的基础问题。

**二、实验内容**

1、应用MATLAB语言编写程序读入一副彩色图像，并分别将其转换为灰度图像和索引色图像。

2、利用imfinfo函数对上述彩色图像进行信息查询。

3、编写MATLAB程序，显示不同类型的图像文件。

4、对1中的灰度图像，进行如下几何变换：

1. 向右平移50个像素；
2. X放大2.5倍，Y放大1.5倍；
3. 顺时针旋转45度，并给出最近邻插值、双线性插值的旋转结果；
4. 给出图像的镜像结果；
5. 将图像x坐标和y坐标进行互换，得到转置后的结果。

**三、实验结果（每个内容包含程序和结果图两个部分）**

1. RGB=imread('头像.jpg');

2. figure (1);

3. imshow(RGB);

4. I=rgb2gray(RGB);

5. figure (2);

6. imshow(I);

7. R=grayslice(I,128);

8. figure (3);

9. imshow(R);

10. info=imfinfo('头像.jpg');

11. S1=imtranslate(I,[50, 0],'OutputView','full');

12. figure (4)

13. imshow(S1);

14. [sx,sy]=size(I);

15. sx=sx.\*1.5;

16. sy=sy.\*2.5;

17. S2=imresize(I,[sx sy]);

18. figure (5)

19. imshow(S2);

20. S3=imrotate(I,-45,'nearest','loose');

21. figure (6)

22. imshow(S3);

23. title('×î½üÁÚ²î·¨Ðý×ª');

24. S4=imrotate(I,-45,'bilinear','loose');

25. figure (7)

26. imshow(S4);

27. title('最近邻差法旋转');

28. [ROW,COL,DIM] = size(I);

29. Ih = uint8(zeros(ROW,COL,DIM));

30. for i =1:ROW

31. for j=1:COL

32. for k=1:DIM

33. x = i;

34. y = COL-j+1;

35. z = k;

36. Ih(x,y,z) =I(i,j,k);

37. end

38. end

39. end

40. figure (8)

41. imshow(Ih);

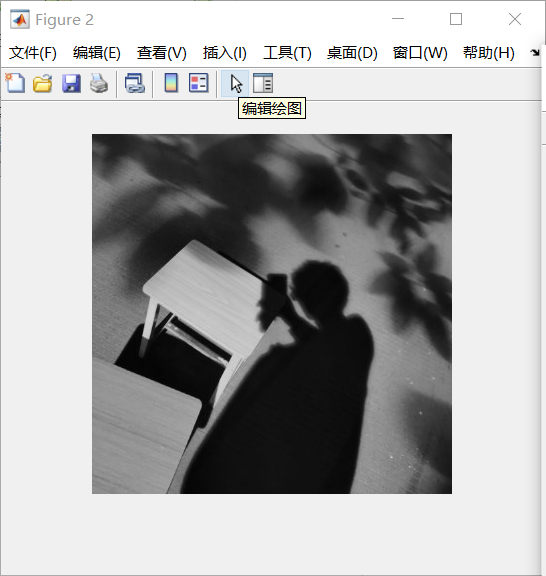
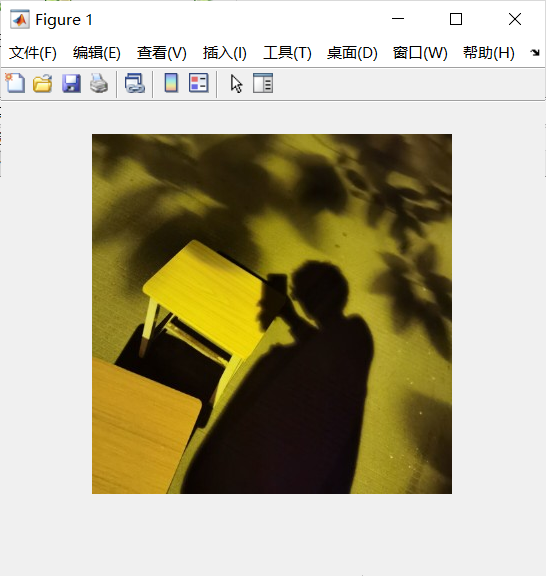
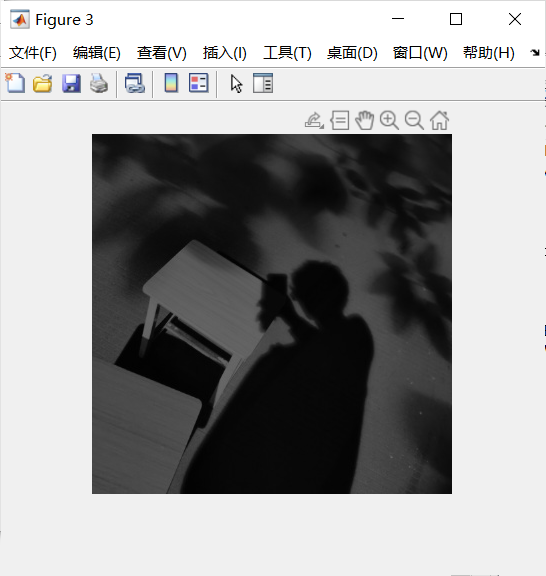
42. title('水平');

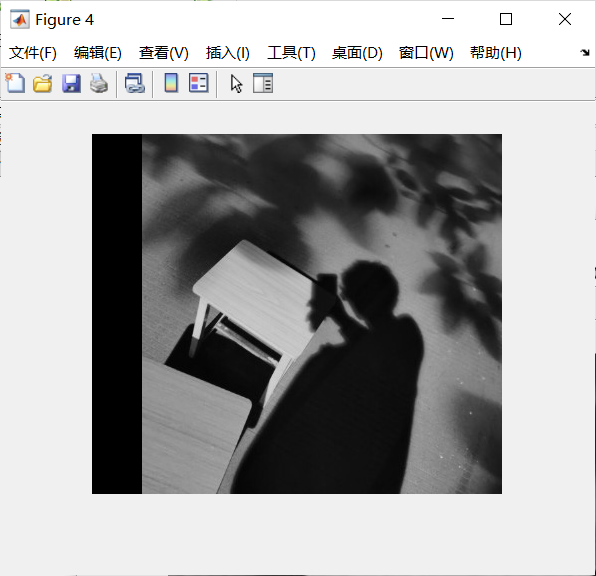
43. S8=I';

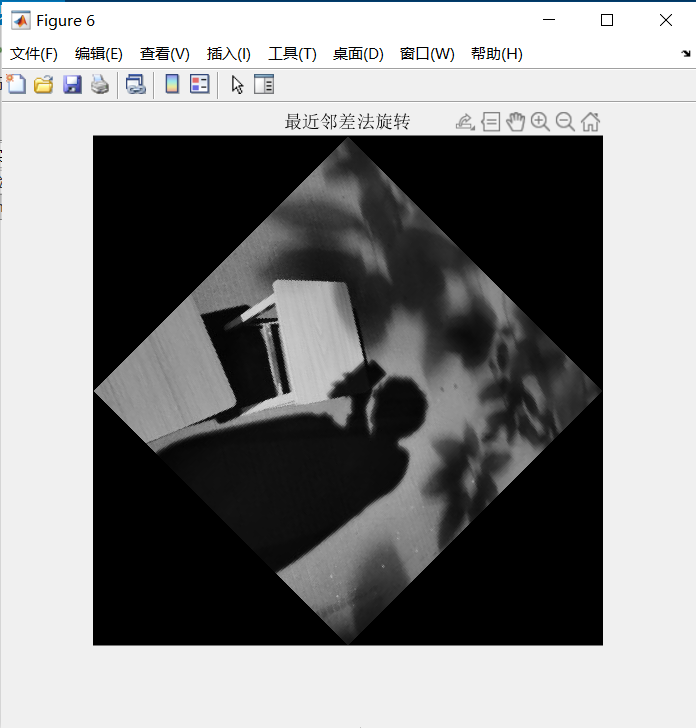
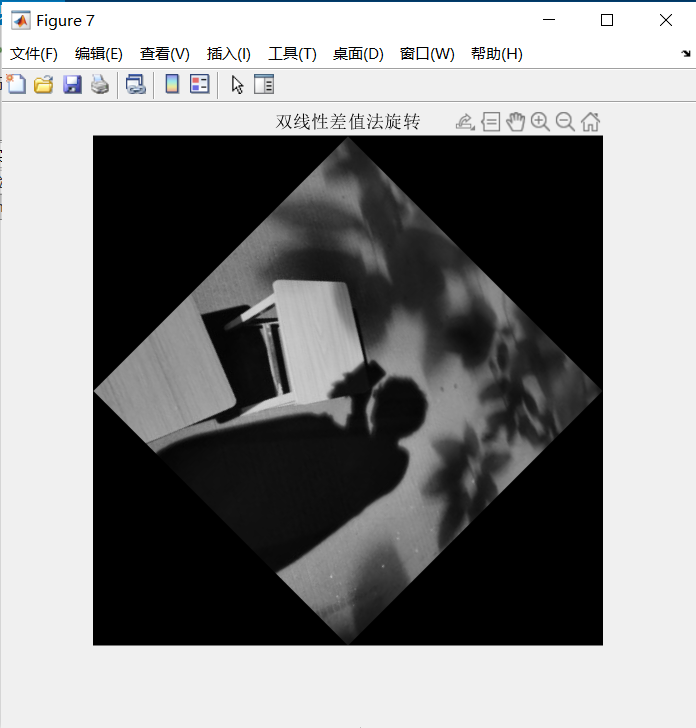
44. figure (9)

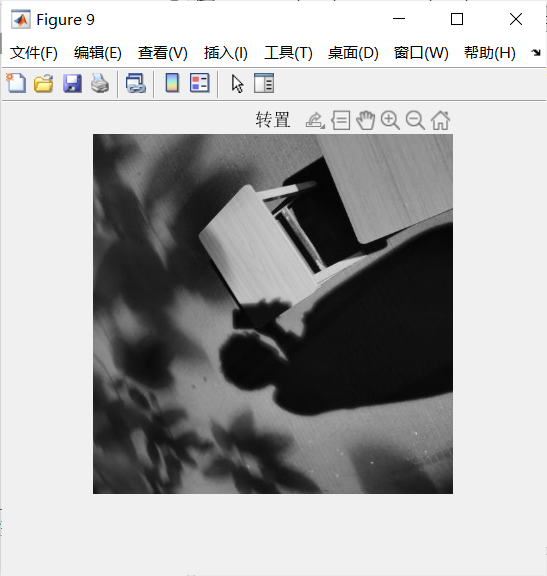
45. imshow(S8);

46. title('转置');

****

****

**** ****

****

**四、实验结果分析**

对于转置操作，我发现了可以直接对矩阵取反来操作；但是对于镜像反转操作，我却没找到更好的解决办法。

**实验二 图像频率域变换**

**一、实验目的**

1.理解频率域的基本原理以及方法；

2.掌握利用MATLAB命令对图像进行傅里叶变换的基本操作；

2.掌握利用MATLAB命令对图像进行余弦变换的基本操作。

**二、实验原理**

1.傅里叶变换

傅里叶变换是能将满足一定条件的某个函数表示成三角函数（正弦或余弦函数）或者它们的积分的线性组合，具有多种不同的变体形式，如连续傅里叶变换和离散傅里叶变换。

二维傅里叶变换的傅里叶变换对:

2.余弦变换

余弦变换是一种正交变换，其变换核为实数的余弦函数。

二维离散余弦变换：

**三、实验内容**

1、在532\*532图像中构造一个长200，宽50的矩形，要求：

1. 在MATLAB中显示上述图像；
2. 显示a）图像对应的傅里叶频谱；
3. 将矩形向左、向右、向上、向下分别移动50个像素，对比傅里叶频谱；
4. 顺时针旋转45度，对比傅里叶频谱；
5. 保持矩形长不变，宽度扩大2倍；保持矩形宽不变，长缩小2倍，对比傅里叶频谱变化。

2、编写MATLAB程序，对一副图像进行二维傅里叶变换，并将其0频率分量移到矩阵的中心。

3、编写MATLAB程序，对一副图像进行二维余弦变换，分别对余弦变换后的余弦系数采用5，15，20，40作为阈值，即低于阈值的赋0，高于00阈值的保留，画出重构后的图像，并计算均方根误差。（提示：参考例3-11）

**三、实验结果**

**1.a**

1. I=zeros(532);

2. for i=166:366

3. for j=241:291

4. I(i,j)=1;

5. end

6. end

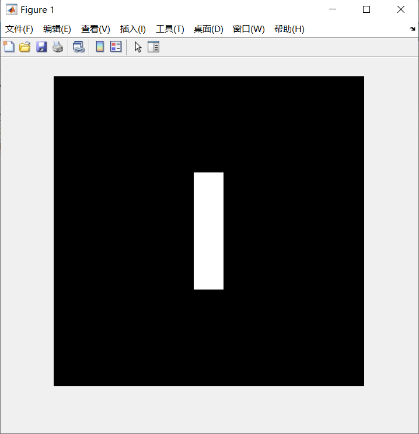
7. J=mat2gray(I);

8. K=im2bw(J,0.5);

9. figure (1);

10. imshow(K);

11.

****

**1.b**

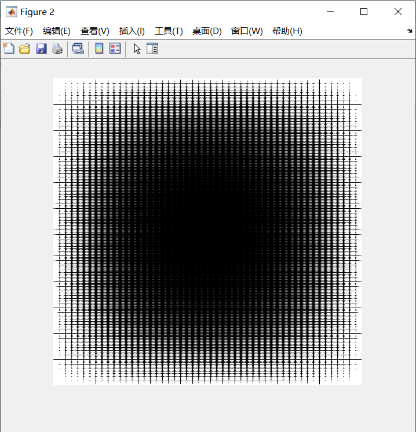
1. G=fft2(J);

2. G1=log(abs(G));

3. figure (2)

4. imshow(G1);

5.

****

**1.c**

1. t=50;

2. I1=zeros(532);

3. I1(166:366,241-t:291-t)=1;

4. figure (1);

5. imshow(I1);

6. G1=fft2(I1);

7. V1=log10(1+abs(G1));

8. figure (2);

9. imshow(V1);

10. I2=zeros(532);

11. I2(166:366,241+t:291+t)=1;

12. figure (3);

13. imshow(I2);

14. G2=fft2(I2);

15. V2=log10(1+abs(G2));

16. figure (4);

17. imshow(V2);

18. I3=zeros(532);

19. I3(166+t:366+t,241:291)=1;

20. figure (5);

21. imshow(I3);

22. G3=fft2(I3);

23. V3=log10(1+abs(G3));

24. figure (6);

25. imshow(V3);

26. I4=zeros(532);

27. I4(166-t:366-t,241:291)=1;

28. figure (7);

29. imshow(I4);

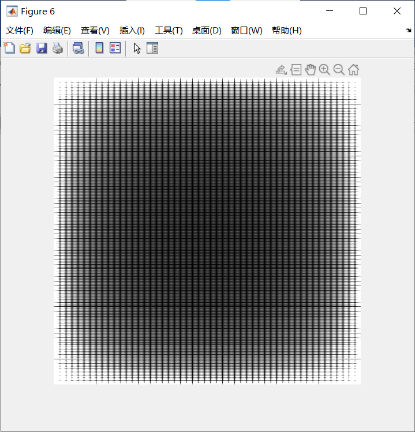
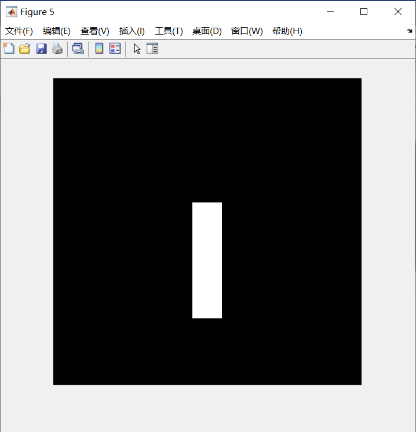
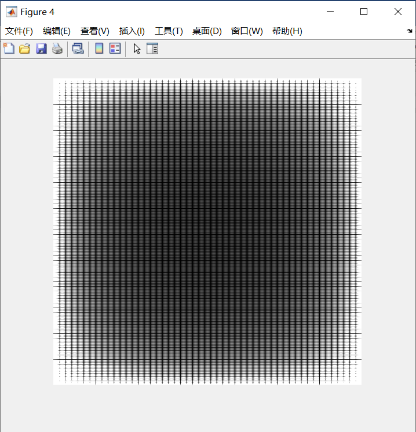
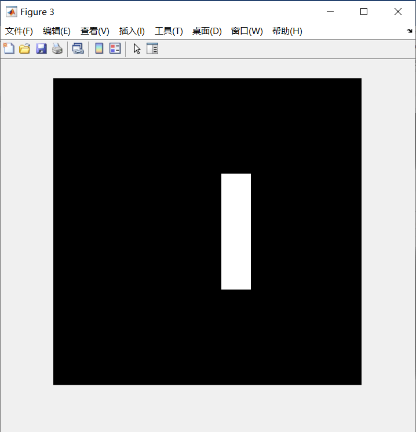
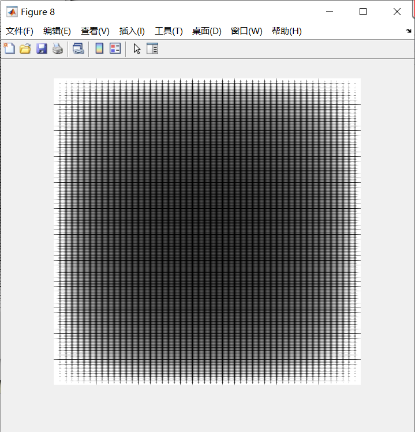
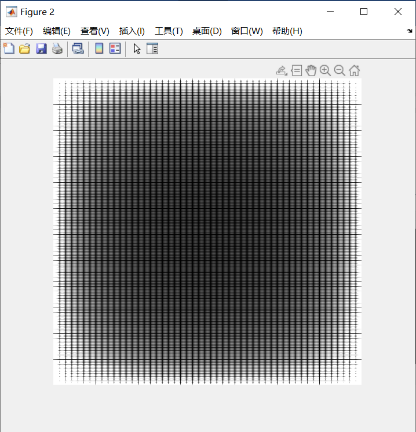
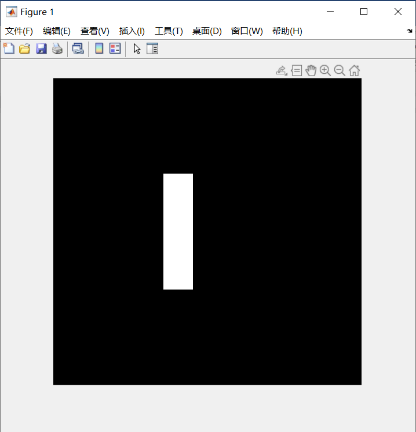
30. G4=fft2(I4);

31. V4=log10(1+abs(G4));

32. figure (8);

33. imshow(V4);

34.

****

**1.d**

1. I=zeros(532);

2. I(166:366,241:291)=1;

3. J=imrotate(I,-45);

4. figure (1);

5. imshow(J);

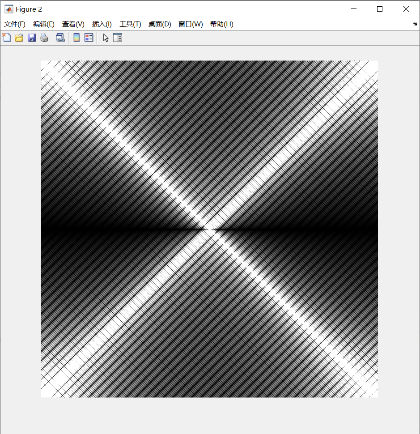
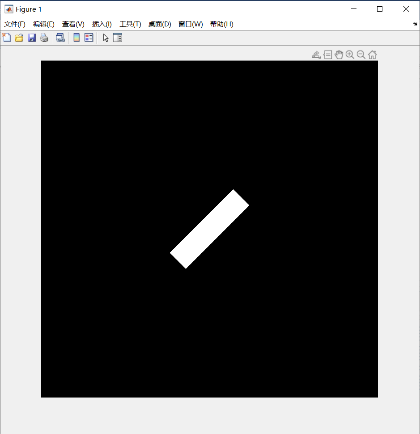
6. G=fft2(J);

7. V=log10(1+abs(G));

8. figure (2);

9. imshow(V);

10.

****

**1.e**

1. I=zeros(532);

2. I(166-100:366+100,241:291)=1;

3. figure (1);

4. imshow(I);

5. G=fft2(I);

6. V=log10(1+abs(G));

7. figure (2);

8. imshow(V);

9. I1=zeros(532);

10. I1(166:366,241-25:291+25)=1;

11. figure (3);

12. imshow(I1);

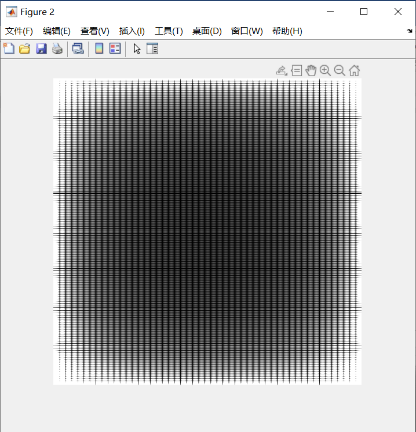
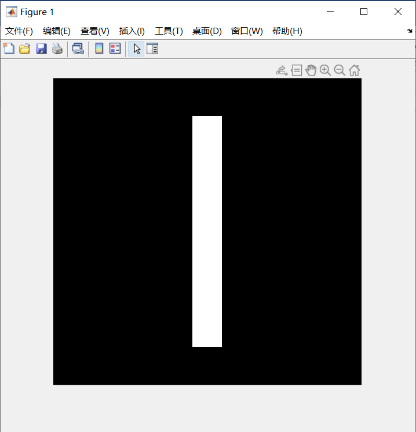
13. G1=fft2(I1);

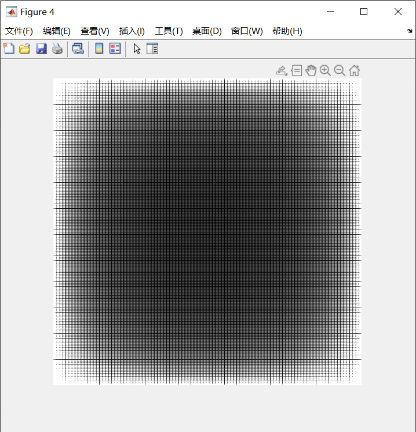
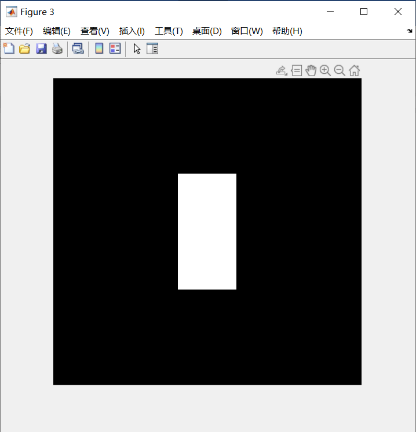
14. V1=log10(1+abs(G1));

15. figure (4);

16. imshow(V1);

17.



****

**2.**

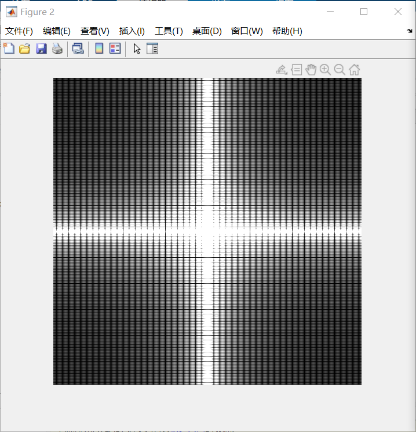
1. G=fftshift(G);

2. V=log10(1+abs(G));

3. figure (2);

4. imshow(V);

5.

****

**3.**

**原图：**



**阈值40：**

1. I=imread("原图.jpg");

2. C=rgb2gray(I);

3. B=dct2(C);

4. figure (1);

5. B(abs(B)<40)=0;

6. K=idct2(B);

7. imshow(K,[]);

8. imwrite(K,'改图.png');

9. a=mean(K(:));

10. M=zeros(360,360);

11. for i=1:360

12. for j=1:360

13. M(i,j)=(K(i,j)-C(i,j)).^2;

14. end

15. end

16. S=(mean(M(:))).^0.5;

17.

均方根误差：S=3.6556

****

**阈值20：**

1. I=imread("原图.jpg");

2. C=rgb2gray(I);

3. B=dct2(C);

4. figure (1);

5. B(abs(B)<20)=0;

6. K=idct2(B);

7. imshow(K,[]);

8. imwrite(K,'改图.png');

9. a=mean(K(:));

10. M=zeros(360,360);

11. for i=1:360

12. for j=1:360

13. M(i,j)=(K(i,j)-C(i,j)).^2;

14. end

15. end

16. S=(mean(M(:))).^0.5;

17.

均方根误差：S=2.7156

****

**阈值15：**

1. I=imread("原图.jpg");

2. C=rgb2gray(I);

3. B=dct2(C);

4. figure (1);

5. B(abs(B)<15)=0;

6. K=idct2(B);

7. imshow(K,[]);

8. imwrite(K,'改图.png');

9. a=mean(K(:));

10. M=zeros(360,360);

11. for i=1:360

12. for j=1:360

13. M(i,j)=(K(i,j)-C(i,j)).^2;

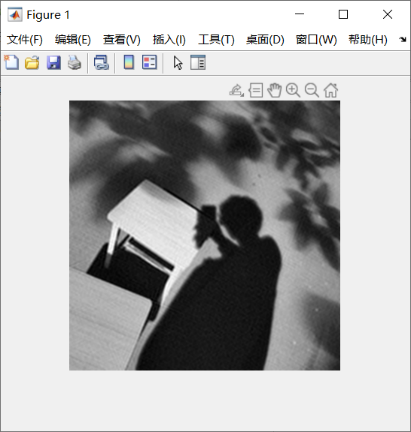
14. end

15. end

16. S=(mean(M(:))).^0.5;

17.

均方根误差：S= 2.3733



**阈值5：**

1. I=imread("原图.jpg");

2. C=rgb2gray(I);

3. B=dct2(C);

4. figure (1);

5. B(abs(B)<5)=0;

6. K=idct2(B);

7. imshow(K,[]);

8. imwrite(K,'改图.png');

9. a=mean(K(:));

10. M=zeros(360,360);

11. for i=1:360

12. for j=1:360

13. M(i,j)=(K(i,j)-C(i,j)).^2;

14. end

15. end

16. S=(mean(M(:))).^0.5;

17.

均方根误差：S=1.3033

****

**四、实验结果分析**

对于实验1.a来说，创建矩阵时考虑到0为黑色，1为白色即可。

对于实验1.b来说，应用对应的函数fft2即可。

对于实验1.c来说，平移部分需要使用到平移函数，也可以重新构建一个图片。之后与1.b实验内容相同。

对于实验1.d来说，使用旋转函数imrotate进行旋转即可。之后与1.b实验内容相同。

对于实验1.e来说，拉伸部分需要使用到拉伸函数，也可以重新构建一个图片。之后与1.b实验内容相同。

对于实验2来说，使用fftshift函数使其0频率分量移到矩阵的中心，之后与1.b实验内容相同。

对于实验3来说，使用dct2函数进余弦变换，之后置零阈值内的数，最后输出图片。均方根误差则使用函数求解。得出结论：阈值设置的越大，均方根误差越大。

**实验三 图像增强**

**一、实验目的**

1、理解直方图均衡化的原理，掌握在MATLAB中对直方图均衡化的应用；

2、理解空间域滤波的原理，掌握MATLAB中对图像进行邻域平均滤波、中值滤波的方法；

3、理解频率域滤波的原理，掌握MATLAB中对图像进行频率域滤波的方法。

**二、实验原理**

1.直方图均衡化

将图像做适当的调整，实现使图像清晰的目的。

2.邻域平均法和中值滤波

邻域平均法：用几个像素灰度的平均值来代替每个像素的灰度。

中值滤波：用一个含有奇数个像素的滑动窗口，将窗口正中点的灰度值用窗口内各点的中值代替。

3.频率域滤波：

将图像从空间域变为频率域（傅里叶变换）。

高斯滤波

是一种线性平滑滤波，适用于消除高斯噪声，广泛应用于图像处理的减噪过程。

Sobel滤波：

Sobel滤波可以提取特定方向的边缘。

Laplacian滤波：

Laplacian滤波是对图像亮度进行二次微分从而检测边缘的滤波。

**三、实验内容**

1、选取一副彩色图像，转换为灰度图像后，对其进行灰度变换：

1. 将灰度图进行归一化处理，并将灰度范围拓展为[0,1]；
2. 将a）图像灰度截取为[0.2,0.8]进行显示；
3. 对b)图像进行全域线性变换，转换为灰度范围为[0,1]的图像；
4. 对c)图像进行分段线性变换，参考教材91页公式(4.2-5)，其中a=0.2, b=0.8, c=0.4, d=0.6,显示灰度变换后的结果；
5. 对c)图像非线性灰度变换，设新的灰度值为*g*(*x*,*y*)，c)图像的灰度值为*f*(*x*,*y*),令*g*(*x*,*y*)=log2(*f*(*x*,*y*)+1)，显示非线性灰度变换后的结果。

2、另外选取一副彩色图像，转换为灰度图像后，对其进行直方图修正：

1. 画出直方图；
2. 对a）中直方图进行均衡化处理，并对比直方图均衡化后的图像和原图像的区别；
3. 对a）中直方图进行规定化处理，使得直方图服从对数分布，对比直方图规定化后的图像和原图像的区别；

3、任取一副灰度图像，对其添加高斯噪声、瑞利噪声、伽马噪声、指数噪声、均匀噪声和脉冲噪声，比较空域滤波和频率域滤波对噪声的平滑效果。

1. 空域滤波至少包括**邻域平均法**和**中值滤波法**；
2. 频率域滤波至少包括**高斯滤波**、**sobel**、**laplacian**；

**三、实验结果**

**1.a**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=double(GR);

4. for i=1:360

5. for j=1:360

6. GR(i,j)=GR(i,j)/double(255);

7. end

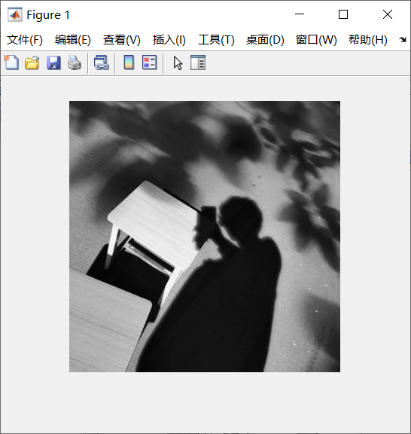
8. end

9. I=mat2gray(GR);

10. figure (1);

11. imshow(I);

12.

****

**1.b**

1. for i=1:360

2. for j=1:360

3. if I(i,j)<=0.2||I(i,j)>=0.8

4. I(i,j)=1;

5. end

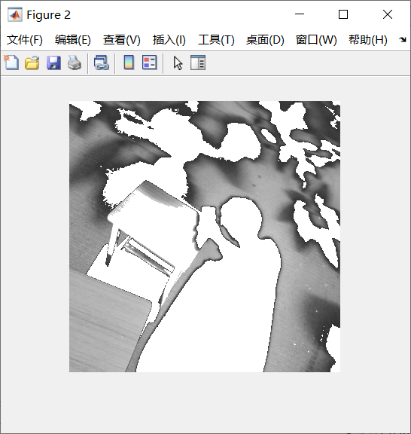
6. end

7. end

8. figure (2);

9. imshow(I);

10.

****

**1.c**

1. for i=1:360

2. for j=1:360

3. I(i,j)=(1/0.6)\*(I(i,j)-0.2);

4. end

5. end

6. figure (3);

7. imshow(I);

8.

****

**1.d**

1. I1=zeros(360);

2. for i=1:360

3. for j=1:360

4. if I(i,j)<=0.2

5. I1(i,j)=2\*I(i,j);

6. elseif I(i,j)>=0.2||I(i,j)<=0.8

7. I1(i,j)=3\*(I(i,j)-0.2)+0.4;

8. else

9. I1(i,j)=2\*(I(i,j)-0.8)+0.6;

10. end

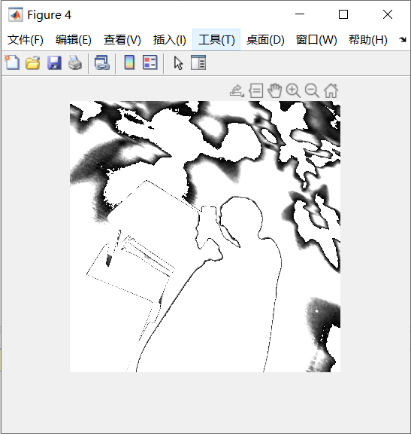
11. end

12. end

13. figure (4);

14. imshow(I1);

15.

****

**1.e**

1. I2=zeros(360);

2. for i=1:360

3. for j=1:360

4. I2(i,j)=log2(I(i,j)+1);

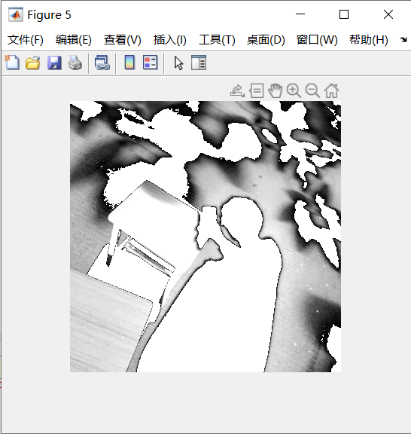
5. end

6. end

7. figure (5);

8. imshow(I2);

9.

****

**2.a**

1. RGB=imread("另一个头像.jpg");

2. GR=rgb2gray(RGB);

3. CS=GR;

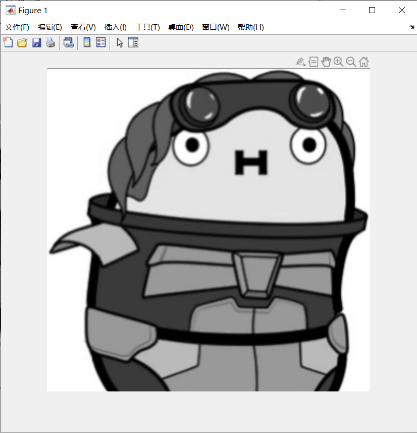
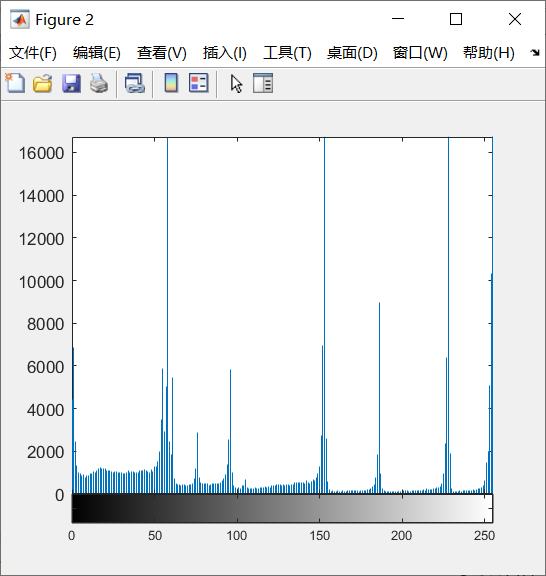
4. figure (1);

5. imshow(GR);

6. figure (2);

7. imhist(GR);

8.

****

**2.b**

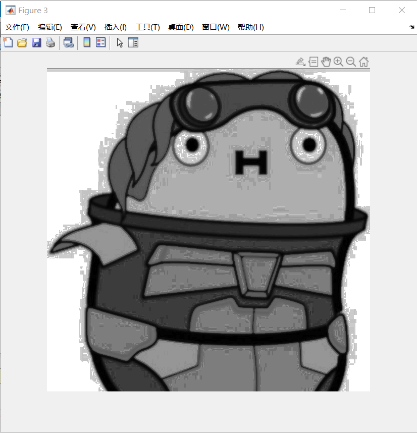
1. GR=im2double(GR);

2. GR=histeq(GR);

3. figure (3);

4. imshow(GR);

5.

****

**2.c**

1. a=0:1:255;

2. a=a/255;

3. s=log2(a+1);

4. S=sum(s);

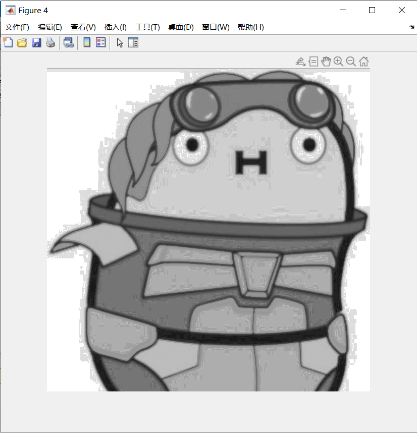
5. h=s/S;

6. GR=histeq(GR,h);

7. figure (4);

8. imshow(GR);

9.

****

**3.a** **高斯噪声**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=im2double(GR);

4. GR=mat2gray(GR);

5. %高斯噪声

6. GR1=imnoise(GR,'gaussian',0,0.02);

7. figure (1);

8. imshow(GR1);

9. %邻域平均法

10. GR11=filter2(fspecial('average',3),GR1);

11. figure (2);

12. imshow(GR11);

13. %中值滤波法

14. GR12=medfilt2(GR1);

15. figure (3);

16. imshow(GR12);

17. %高斯滤波

18. gr13=fspecial('gaussian',4,0.3);

19. GR13=filter2(gr13,GR1,'same');

20. figure (4);

21. imshow(GR13);

22. %sobel滤波

23. gr14=fspecial('sobel');

24. GR14=filter2(gr14,GR1,'same');

25. figure (5);

26. imshow(GR14);

27. %laplacian滤波

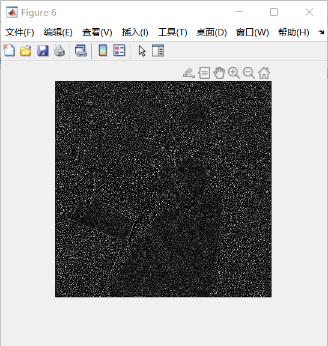
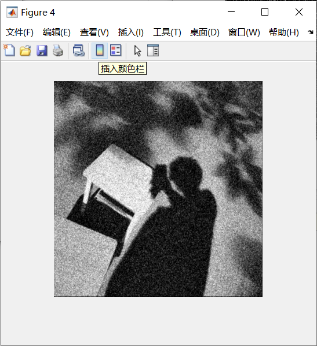
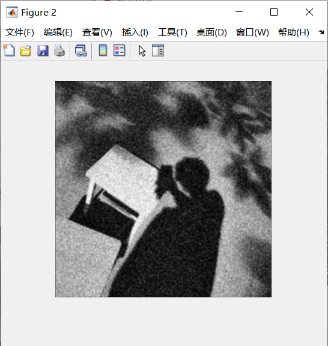
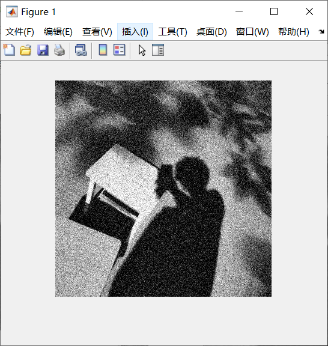
28. gr15=fspecial('laplacian',0.5);

29. GR15=filter2(gr15,GR1,'same');

30. figure (6);

31. imshow(GR15);

32.

****

**3.b** **瑞利噪声**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=im2double(GR);

4. GR=mat2gray(GR);

5. [M,N]=size(GR);

6. %瑞利噪声

7. a=0;

8. b=0.08;

9. B=1;

10. GR20=a+b\*raylrnd(B,M,N);

11. GR2=GR+GR20;

12. figure (1);

13. imshow(GR2);

14. %邻域平均法

15. GR21=filter2(fspecial('average',3),GR2);

16. figure (2);

17. imshow(GR21);

18. %中值滤波法

19. GR22=medfilt2(GR2);

20. figure (3);

21. imshow(GR22);

22. %高斯滤波

23. gr23=fspecial('gaussian',4,0.3);

24. GR23=filter2(gr23,GR2,'same');

25. figure (4);

26. imshow(GR23);

27. %sobel滤波

28. gr24=fspecial('sobel');

29. GR24=filter2(gr24,GR2,'same');

30. figure (5);

31. imshow(GR24);

32. %laplacian滤波

33. gr25=fspecial('laplacian',0.5);

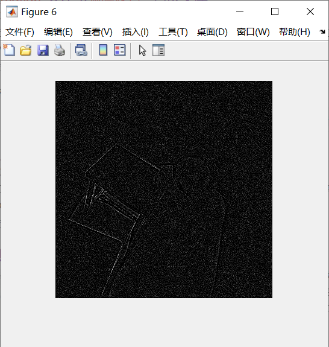
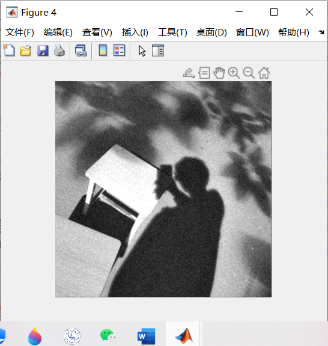
34. GR25=filter2(gr25,GR2,'same');

35. figure (6);

36. imshow(GR25);

37.

****

****

**3.c** **伽马噪声**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=im2double(GR);

4. GR=mat2gray(GR);

5. [M,N]=size(GR);

6. %伽马噪声

7. a=0;

8. b=0.08;

9. A=1;

10. B=2;

11. GR30=a+b\*gamrnd(A,B,[M,N]);

12. GR3=GR+GR30;

13. figure (1);

14. imshow(GR3);

15. %邻域平均法

16. GR31=filter2(fspecial('average',3),GR3);

17. figure (2);

18. imshow(GR31);

19. %中值滤波法

20. GR32=medfilt2(GR3);

21. figure (3);

22. imshow(GR32);

23. %高斯滤波

24. gr33=fspecial('gaussian',4,0.3);

25. GR33=filter2(gr33,GR3,'same');

26. figure (4);

27. imshow(GR33);

28. %sobel滤波

29. gr34=fspecial('sobel');

30. GR34=filter2(gr34,GR3,'same');

31. figure (5);

32. imshow(GR34);

33. %laplacian滤波

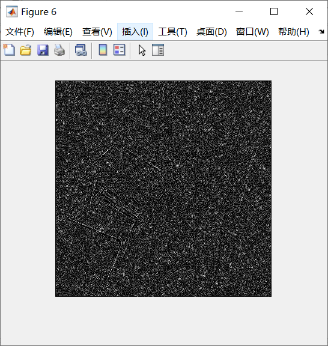
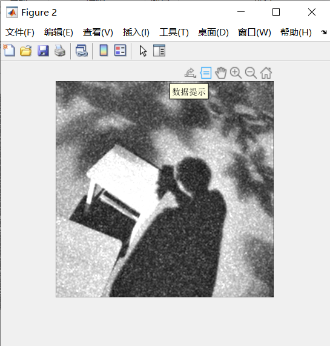
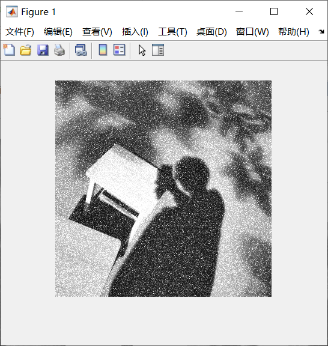
34. gr35=fspecial('laplacian',0.5);

35. GR35=filter2(gr35,GR3,'same');

36. figure (6);

37. imshow(GR35);

38.

****

**3.d** **指数噪声**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=im2double(GR);

4. GR=mat2gray(GR);

5. [M,N]=size(GR);

6. %指数噪声

7. a=0;

8. b=0.08;

9. mu=2;

10. GR40=a+b\*exprnd(mu,[M,N]);

11. GR4=GR+GR40;

12. figure (1);

13. imshow(GR4);

14. %邻域平均法

15. GR41=filter2(fspecial('average',3),GR4);

16. figure (2);

17. imshow(GR41);

18. %中值滤波法

19. GR42=medfilt2(GR4);

20. figure (3);

21. imshow(GR42);

22. %高斯滤波

23. gr43=fspecial('gaussian',4,0.3);

24. GR43=filter2(gr43,GR4,'same');

25. figure (4);

26. imshow(GR43);

27. %sobel滤波

28. gr44=fspecial('sobel');

29. GR44=filter2(gr44,GR4,'same');

30. figure (5);

31. imshow(GR44);

32. %laplacian滤波

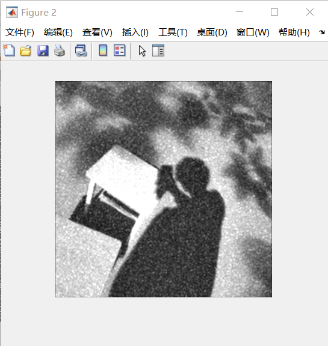
33. gr45=fspecial('laplacian',0.5);

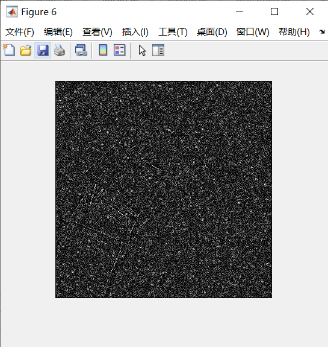
34. GR45=filter2(gr45,GR4,'same');

35. figure (6);

36. imshow(GR45);

37.

****

****

**3.e** **均匀噪声**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=im2double(GR);

4. GR=mat2gray(GR);

5. [M,N]=size(GR);

6. %均匀噪声

7. a=0;

8. b=0.08;

9. A=0;

10. B=2;

11. GR50=a+b\*unifrnd(A,B,[M,N]);

12. GR5=GR+GR50;

13. figure (1);

14. imshow(GR5);

15. %邻域平均法

16. GR51=filter2(fspecial('average',3),GR5);

17. figure (2);

18. imshow(GR51);

19. %中值滤波法

20. GR52=medfilt2(GR5);

21. figure (3);

22. imshow(GR52);

23. %高斯滤波

24. gr53=fspecial('gaussian',4,0.3);

25. GR53=filter2(gr53,GR5,'same');

26. figure (4);

27. imshow(GR53);

28. %sobel滤波

29. gr54=fspecial('sobel');

30. GR54=filter2(gr54,GR5,'same');

31. figure (5);

32. imshow(GR54);

33. %laplacian滤波

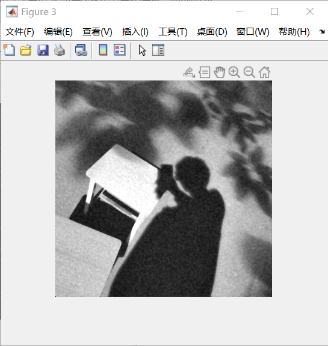
34. gr55=fspecial('laplacian',0.5);

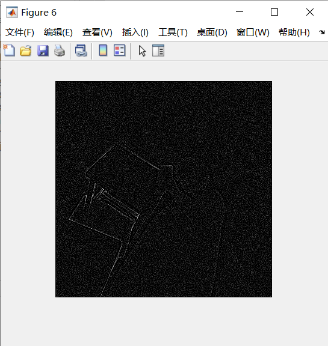
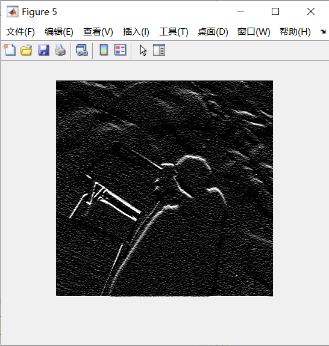
35. GR55=filter2(gr55,GR5,'same');

36. figure (6);

37. imshow(GR55);

38.

****

****

**3.f 脉冲噪声**

1. RGB=imread("头像.jpg");

2. GR=rgb2gray(RGB);

3. GR=im2double(GR);

4. GR=mat2gray(GR);

5. %脉冲噪声

6. GR6=imnoise(GR,'salt & pepper',0.02);

7. figure (1);

8. imshow(GR6);

9. %邻域平均法

10. GR61=filter2(fspecial('average',3),GR6);

11. figure (2);

12. imshow(GR61);

13. %中值滤波法

14. GR62=medfilt2(GR6);

15. figure (3);

16. imshow(GR62);

17. %高斯滤波

18. gr63=fspecial('gaussian',4,0.3);

19. GR63=filter2(gr63,GR6,'same');

20. figure (4);

21. imshow(GR63);

22. %sobel滤波

23. gr64=fspecial('sobel');

24. GR64=filter2(gr64,GR6,'same');

25. figure (5);

26. imshow(GR64);

27. %laplacian滤波

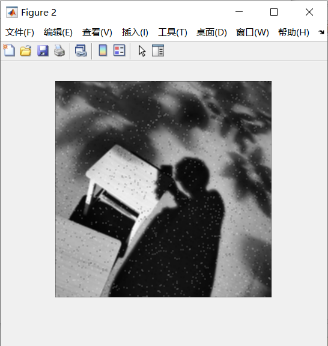
28. gr65=fspecial('laplacian',0.5);

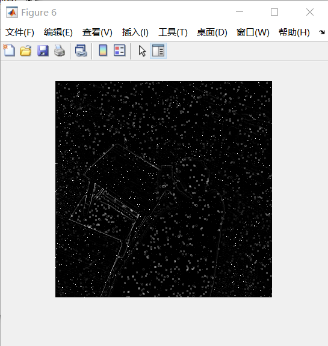
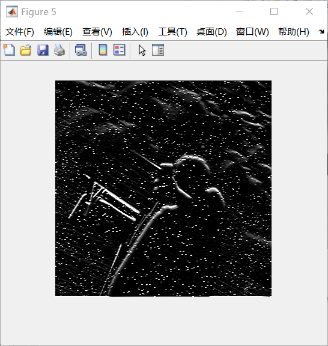
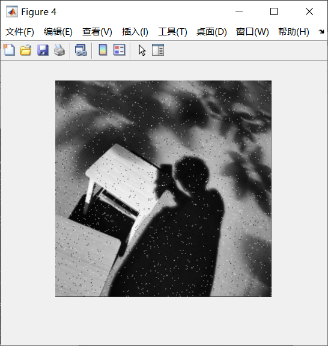
29. GR65=filter2(gr65,GR6,'same');

30. figure (6);

31. imshow(GR65);

32.





**四、实验结果分析**

1.a：归一化处理意思是将灰度图像的每一个像素点取值范围由0-255变为0-1；所用double函数可以将u8类型数据转换为double型。而后我又发现可以用im2double函数直接转换以及缩小范围。而后我打算全部换成im2double函数。而mat2gray函数可以将归一化之后灰度图显示出来。

1.b：范围显示比较简单，将范围内的保留，范围外的置空即可。

1.c：全域线性变换则需要参考公式，将公式写出来即可。

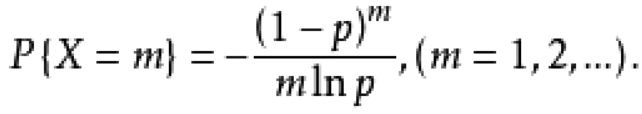
1.d：分段线性变换则需要把所用到的公式分段描述，表达之后即可。

1.e：非线性灰度转换可以参考其公式，将每个像素点进行处理即可。

2.a：绘制直方图使用imhist函数即可。

2.b：均衡化使用histeq函数即可。

2.c：规定化函数书上并未说明，则需要自行写出。先找到对数分布的公式如下：



再将公式内的式子计算出来，将结果填到histeq的第二个参数中即可。

3.a：高斯噪声的添加方式为使用imnoise函数，在第二个参数中填入“gaussian”即可。邻域平均法使用filter2函数即可；中值滤波法使用medfilt2函数即可。高斯滤波使用fspecial函数，中间参数填入gaussian，再使用filter2函数即可。Sobel滤波则使用fspecial函数，中间参数填入sobel，再使用filter2函数即可。Laplacian滤波使用fspecial函数，中间参数填入laplacian，再使用filter2函数即可。

3.b：瑞利噪声没有固定的函数，需要自行构建。使用raylrnd函数构建一个像素矩阵，经过相关运算后得到的像素矩阵与原矩阵相加即可得到瑞利噪声的图像。添加其余函数与上述3.a方法相同。

3.c：伽马噪声也没有固定的函数，将3.b中的raylrnd函数替换为gamrnd函数，再进行相关运算叠加即可。添加其余函数与上述3.a方法相同。

3.d：指数噪声则使用的是exprnd函数。

3.e：均匀噪声则使用的是unifrnd函数。

3.f：脉冲噪声就是椒盐噪声，方法和高斯噪声相同，只是将imnoise中的参数替换为“salt & noise”，其余同上。

**实验四 图像压缩编码**

**一、实验目的**

1、理解霍夫曼编码、傅里叶变换编码以及余弦变换压缩编码的原理；

2、掌握MATLAB中利用霍夫曼编码、傅里叶变换编码以及余弦变换进行压缩编码的方法。

**二、实验原理**

1、霍夫曼编码

2、傅里叶变换编码

3、余弦变换编码

**三、实验内容**

1、选取一副彩色图像，转换为灰度图像后，对其进行霍夫曼编码，给出每个像素的编码结果。

2、假设一个文件有18种符号，分别定义为a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a17 a18，对应的概率为0.05 0.03 0.07 0.12 0.23 0.02 0.07 0.09 0.01 0.02 0.04 0.07 0.02 0.03 0.04 0.06 0.015 0.015，计算采用霍夫曼编码的编码效率。

3、选取一副灰度图像，对其进行变换编码：

1. 比较不同压缩比（2:1 4:1 8:1），采用傅里叶变换编码和余弦变换编码的结果，并给出均方根误差。
2. 在相同压缩比情况下，对比2×2、4×4、8×8、16×16不同分割方式余弦变换编码的结果，并给出均方根误差。

**四、实验结果和分析**

1.

1. clear,clc;

2. I1=imread("Í·Ïñ.jpg");

3. I2=rgb2gray(I1);

4. imshow(I2);

5. [R,C]=size(I2);

6. a=I2(:);

7. P=zeros(1,256);

8. for i=0:255

9. P(i+1)=length(find(a==i))/(R\*C);

10. end

11. [output,p]=hufuman(I2);

12. H=0;

13. for i=1:256

14. if p(i) ~= 0

15. H=H+p(i)\*log2(1/p(i));

16. end

17. end

18. R=0;

19. for i=1:256

20. R=length(strtrim(output(i,:)))\*p(i)+R;

21. end

22. n=1-(H/(R\*log2(256)));

23. fprintf('ÈßÓà¶ÈnÎª %f\n',n )

24. output;

25. sum(p)

26. function [output,p]=hufuman(input)

27. [R,C]=size(input);

28. a=input(:);

29. p=zeros(1,256);

30. for i=0:255

31. p(i+1)=length(find(a==i))/(R\*C);

32. end

33. n=length(p);

34. p=sort(p);

35. q=p;

36. m=zeros(n-1,n);

37. for i=1:n-1

38. [q,e]=sort(q);

39. m(i,:)=[e(1:n-i+1),zeros(1,i-1)];

40. q=[q(1)+q(2),q(3:n),1];

41. end

42.

43. for i=1:n-1

44. c(i,1:n\*n)=blanks(n\*n);

45. end

46. c(n-1,n)='1';

47. c(n-1,2\*n)='0';

48. for i=2:n-1

49. c(n-i,1:n-1)=c(n-i+1,n\*(find(m(n-i+1,:)==1))-(n-2):n\*(find(m(n-i+1,:)==1)));

50. c(n-i,n)='0';

51. c(n-i,n+1:2\*n-1)=c(n-i,1:n-1);

52. c(n-i,2\*n)='1';

53. for j=1:i-1

54. c(n-i,(j+1)\*n+1:(j+2)\*n)=c(n-i+1,n\*(find(m(n-i+1,:)==j+1)-1)+1:n\*find(m(n-i+1,:)==j+1));

55. end

56. end

57. for i=1:n

58. output(i,1:n)=c(1,n\*(find(m(1,:)==i)-1)+1:find(m(1,:)==i)\*n);

59. len(i)=length(find(abs(output(i,:))~=32));

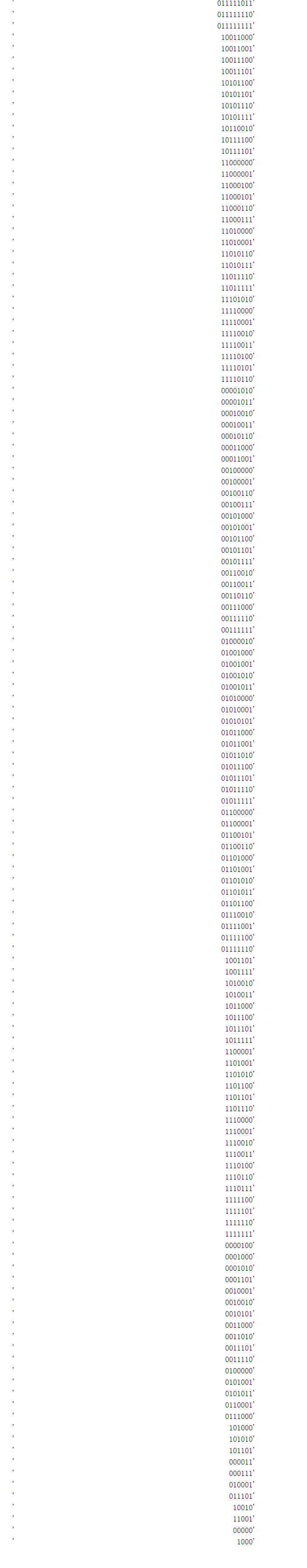
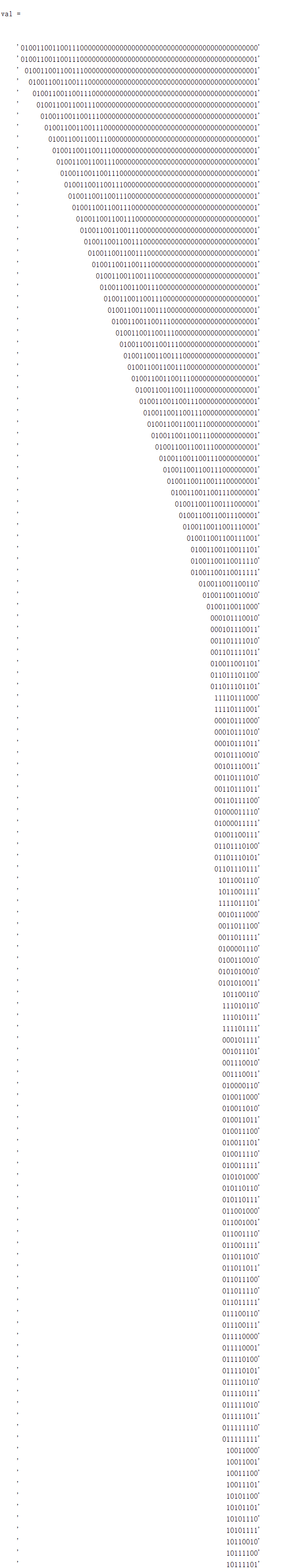
60. end

61. output=strtrim(output);

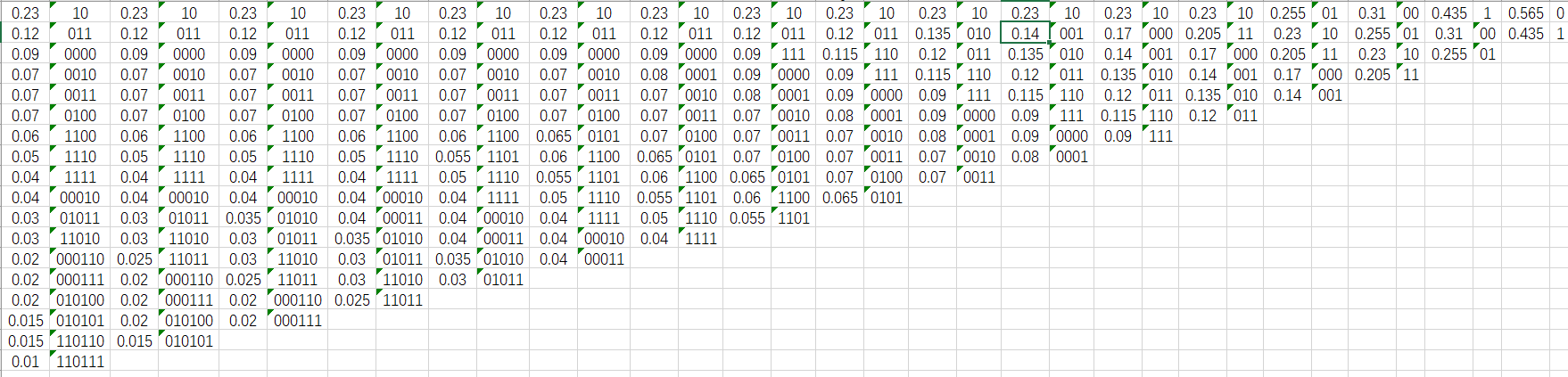
62. end

63.

结果：



2.我运用excel完成了霍夫曼编码，截图如下：



最终得出各个概率对应的码长，编写程序得以计算：

1. clc;

2. clear;

3. a=[0.23 2

4. 0.12 3

5. 0.09 4

6. 0.07 4

7. 0.07 4

8. 0.07 4

9. 0.06 4

10. 0.05 4

11. 0.04 4

12. 0.04 5

13. 0.03 5

14. 0.03 5

15. 0.02 6

16. 0.02 6

17. 0.02 6

18. 0.015 6

19. 0.015 6

20. 0.01 6

21. ];

22. a=double(a);

23. b=zeros(1,18);

24. for i=1:18

25. b(i)=a(i,1)\*log2(a(i,1));

26. end

27. H=-sum(b);

28. r=zeros(1,18);

29. n=H/3.72;

30.

结果：编码效率n=99.32%

3.

1. I=imread("头像.jpg");

2. I=rgb2gray(I);

3. I=double(I)/255;

4. [M1,M2]=size(I);

5. K1=zeros(1,M2);

6. for i=1:M2

7. for j=1:M1

8. K1(1,i)=K1(1,i)+I(j,i);

9. end

10. end

11. T=dctmtx(8);

12. T1=fft2(I);

13. B=blkproc(I,[8 8],'P1\*x\*P2',T,T');

14. M=zeros(8,8);

15. for i=1:4

16. for j=1:8

17. M(i,j)=1;

18. end

19. end

20. B2=blkproc(B,[8 8],'P1.\*x',M);

21. I2=blkproc(B2,[8 8],'P1\*x\*P2',T',T);

22. subplot(321);imshow(I2);title('ÓàÏÒ±ä»»2:1');

23. K2=zeros(1,M2);

24. for i=1:M2

25. for j=1:M1

26. K2(1,i)=K2(1,i)+I2(j,i);

27. end

28. end

29. s1=std((K1-K2),1);

30. [k,p]=size(I);

31. t\_fft=blkproc(I,[8 8],'fft2(x)');

32. t\_block=im2col(t\_fft,[8 8],'distinct');

33. [t\_change,ix]=sort(t\_block);

34. [m,n]=size(t\_block);

35. cr=0.5;

36. nums=64\*cr;

37. for i=1:n

38. t\_block(ix(1:nums),i)=0;

39. end

40. t\_rchange=col2im(t\_block,[8 8],[k p],'distinct');

41. t\_ifft=blkproc(t\_rchange,[8 8],'ifft2(x)');

42. subplot(322);imshow(t\_ifft);title('¸µÀïÒ¶2:1');

43. K2=zeros(1,M2);

44. for i=1:M2

45. for j=1:M1

46. K2(1,i)=K2(1,i)+t\_ifft(j,i);

47. end

48. end

49. s2=std((K1-K2),1)

50. for i=1:4

51. for j=1:4

52. M(i,j)=1;

53. end

54. end

55. B2=blkproc(B,[8 8],'P1.\*x',M);

56. I2=blkproc(B2,[8 8],'P1\*x\*P2',T',T);

57. subplot(323);imshow(I2);title('4:1');

58. K2=zeros(1,M2);

59. for i=1:M2

60. for j=1:M1

61. K2(1,i)=K2(1,i)+I2(j,i);

62. end

63. end

64. s3=std((K1-K2),1)

65. [k,p]=size(I);

66. t\_fft=blkproc(I,[8 8],'fft2(x)');

67. t\_block=im2col(t\_fft,[8 8],'distinct');

68. [t\_change,ix]=sort(t\_block);

69. [m,n]=size(t\_block);

70. cr=0.25;

71. nums=64\*cr;

72. for i=1:n

73. t\_block(ix(1:nums),i)=0;

74. end

75. t\_rchange=col2im(t\_block,[8 8],[k p],'distinct');

76. t\_ifft=blkproc(t\_rchange,[8 8],'ifft2(x)');

77. subplot(324);imshow(t\_ifft);title('4:1');

78. K2=zeros(1,M2);

79. for i=1:M2

80. for j=1:M1

81. K2(1,i)=K2(1,i)+t\_ifft(j,i);

82. end

83. end

84. s4=std((K1-K2),1)

85. for i=1:2

86. for j=1:4

87. M(i,j)=1;

88. end

89. end

90. B2=blkproc(B,[8 8],'P1.\*x',M);

91. I2=blkproc(B2,[8 8],'P1\*x\*P2',T',T);

92. subplot(325);imshow(I2);title('8:1');

93. K2=zeros(1,M2);

94. for i=1:M2

95. for j=1:M1

96. K2(1,i)=K2(1,i)+I2(j,i);

97. end

98. end

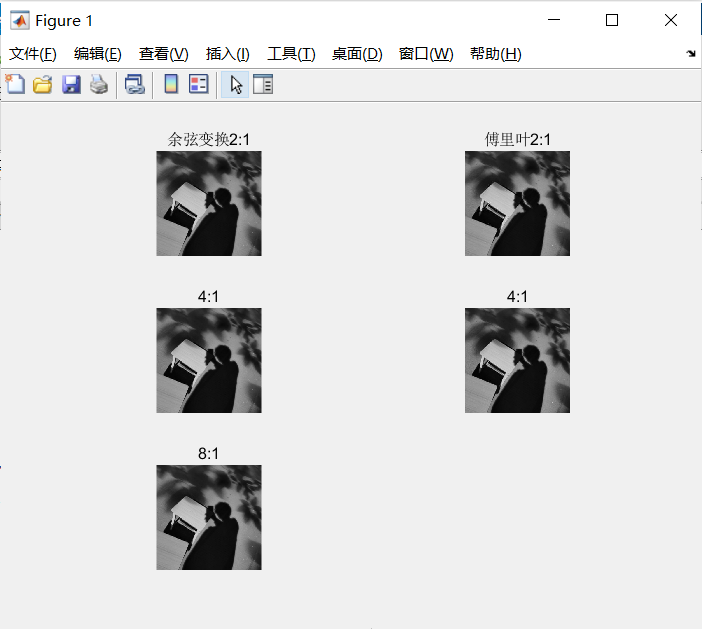
99. s5=std((K1-K2),1)

100. [k,p]=size(I);

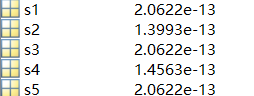
101. t\_fft=blkproc(I,[8 8]);

102. 最终结果：

103.



方均根误差：



|  |  |
| --- | --- |
| **学生实验 心得** | 首先通过本次实验课，我熟练地掌握了matlab的一些基本语法，能用matlab进行简单的编程以及图像处理。同时，我也了解到了一些图像处理的基本思想思路，学会了很多计算技巧。在我一筹莫展之际，老师的辅导以及网上的资源也必不可少。熟练的寻找资源也是一名优秀的大学生的必修课。  其次，每一个实验我都认真完成，代码绝大部分为自己手敲。最后得出正确的结果时，我会有一种成就感和兴奋感。这也是我长期以来手写代码的最佳动力。  最后要感谢我的老师和与我一起写程序的同学们，你们给予了我相当程度上的帮助，谢谢大家，祝愿大家越来越好。  学生（签名）：  2022 年 6月 2 日 |
| **指导**  **教师**  **评语** | 成绩评定：  指导教师（签名）：  年 月 日 |