大连理工大学

本科实验报告

课程名称：多媒体技术

学院（系）：电子信息与电气工程学部

专　　业：电子信息工程

班　　级：电信 1806

学　　号：201871080

学生姓名：刘祎铭

2021 年　10 月 7 日

实验项目列表

序号	实验项目名称	学时	成　　绩（总分 10 分）	
			总分	实际得分
1	数字音频基础及采样频率转换	3	3	
2	彩色数字图像及视频基础	3	3	
3	图像压缩	6	4	
总计	学分：0.5	12	10	

实验一 数字音频基础及采样频率转换

一、 实验目的和要求

1. 了解数字音频资源的常用格式
2. 学会数字音频资源的获取方法
3. 掌握采样频率转换基本原理
4. 用 Matlab 实现简单的采样频率转换

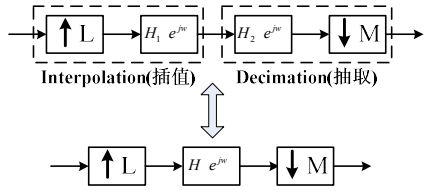
二、 实验原理和内容

采样频率转换(SRC)就是将数字信号从一个采样频率转换到另一个采样频率。从标准音频格式的角度来说，存在着很多的音频标准格式：如 CD 格式，DAT 格式，MP3 格式等。由于这些格式具有不同的采样频率，所以要进行格式转换时必然需要进行采样频率转换。还有 intel AC'97 规范约定了声卡需要经过一个处理过程，即将所有信号重新转换成统一的采样率输出，因此也需要进行采样频率转换。SRC 如果进行了非整数倍的转换的话，比如 44.1KHz 向 48KHz 转换时，会有较大的噪声或者谐波出现，这些噪声因转换器的质量高低，算法好坏而定，不优秀的算法会严重的影响听感。

基于整数倍升采样和整数倍降采样的采样率转换：

（1）单级采样频率转换器：

转换率： $R = L/M$

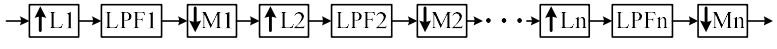


$$H_1(e^{jw}) = \begin{cases} L, & 0 \# |w| \quad \frac{p}{L} \\ 0, & else \end{cases}$$ 防镜像滤波器

$$H_2(e^{jw}) = \begin{cases} 1, & 0 \# |w| \quad \frac{p}{M} \\ 0, & else \end{cases}$$ 抗混叠滤波器

$$H(e^{jw}) = H_1(e^{jw})H_2(e^{jw}) = \begin{cases} L, & 0 \# |w| \quad \min(\text{缺失}, \frac{p}{M}) \\ 0, & else \end{cases}$$

（2）多级采样频率转换器：



$$R = \frac{L}{M} = \frac{L1}{M1}\text{g}\frac{L2}{M2}\ldots\frac{Ln}{Mn}$$

一般要求：$\frac{L1}{M1} > \frac{L2}{M2} > \ldots > \frac{Ln}{Mn}$

基于离散余弦变换的采样频率转换：

（1）离散余弦变换（DCT）：

正变换：$X(k)=w(k)\sum_{n=0}^{N-1}x(n)\cos\frac{(2n+1)kp}{2N},\quad k=0,1,...,N-1$

反变换：$x(n)=\sum_{k=0}^{N-1}w(k)X(k)\cos\frac{(2n+1)kp}{2N},\quad n=0,1,...,N-1$

$$w(k)=\begin{cases}\frac{1}{\sqrt{N}}, & k=0\\ \sqrt{\frac{2}{N}}, & 1\#k\quad N-1\end{cases}$$

（2）基于离散余弦变换（DCT）的采样频率转换：

f_{si}：输入采样率；

f_{so}：输出采样率；

采样率转换：$f_{so}=af_{si}$；$a>1$，称为升采样；$0<a<1$，称为降采样；

原始信号：$x(n)$：$0\#n\quad N$

1) 如果$a>1$,那些转换后新的采样率下的信号就为：$x(m)$：$0\#m\quad aN$；转换过程如下：

[1] $y(k)=\mathrm{DCT}(x(n)),\quad 0\#n\quad N$

[2] $Y(k)=\begin{cases}y(k), & 0\#k\quad N\\ 0, & N<k<PN\end{cases}$ (Appending zero)

[3] $x(m)=\sqrt{a}\,\mathrm{gIDCT}(Y(k)),\quad 0\#m\quad aN$

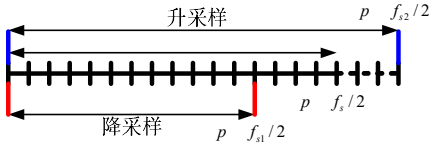
2) 如果$0<a<1$,那些转换后新的采样率下的信号就为：$x(m)$：$0\#m\quad aN$；转换过程如下：

[1] $y(k)=\mathrm{DCT}(x(n)),\quad 0\#n\quad N$

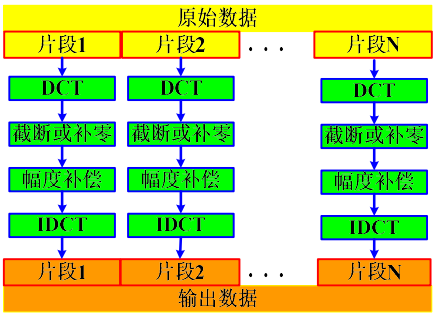
[2] $Y(k)=y(k),\ 0\#k\quad aN\quad (\mathrm{truncation})$

[3] $x(m)=\sqrt{a}\,\mathrm{gIDCT}(Y(k)),\quad 0\#m\quad aN$

（3）基于离散余弦变换（DCT）的采样频率转换原理：



（4）基于离散余弦变换（DCT）的采样频率转换具体实现：



利用离散余弦变换实现音阶变换：

进行采样率变换，然后按原采样率播放

音阶	转换率
-2	$2^{-2/12}$
-1	$2^{-1/12}$
0	1.0
+1	$2^{1/12}$
+2	$2^{2/12}$

三、 主要仪器设备（软件环境）

MATLAB R2019b

四、 实验步骤与操作方法

1. 利用升采样降采样的方式实现 44.1kHz 到 48kHz 的采样频率转换；

首先在信号中插入 0 值，经过低通滤波器滤波以消除镜像；接着需要一个低通滤波器首先对输入信号进行预处理，将高频分量滤除，再进行采样。

代码如下：

```
fileName = 'I could be the one';
[ data, fs   ] = audioread([ fileName '.wav' ]);
t = 10;   %%取前  10s
oldData = data(1:t*fs,:);
%设计滤波器
n=-100:1:100;
filter=sin(pi/150*n)./(pi/150*n);
filter(1,101)=1;
%%采样率转换：
newfs = 48000;
[L,M] = rat(newfs/fs);
out1 = myupfirdn(oldData(:,1), filter, L, M);
out2 =myupfirdn(oldData(:,2), filter, L, M);
out=[out1',out2'];
audiowrite('I could be the one new.wav',out,newfs);
```

myupfirdn 函数：

```
function out = myupfirdn(in, filter, L, M)
%%单级 SRC 的直接实现，类比 matlab upfirdn(多相结构)
out1 = myupsample(in, L);    %%up sample
out2 = conv(out1, filter);       %%low pass filter
out   = mydownsample(out2, M);  %%down sample
function out = myupsample(in, L)
out = zeros(1, (length(in)-1)*L+1);
for j = 1:length(in)
out(L*(j-1)+1) = in(j);
end
function out = mydownsample(in, M)
out = in(1:M:floor((length(in)-1)/M)*M+1);
```

2. 利用离散余弦变换的方式实现 44.1kHz 到 48kHz 的采样频率转换；

```
function [ out ,realRate ] = src_dct(in, rate, bufferSize)
if  rate == 1
    out = in;
    realRate = 1;
end
inputLength  = length(in);  %%输入信号长度
inFragLength = bufferSize;  %%输入信号分段长度
fragNum = floor(inputLength/bufferSize)+1;  %%输入信号分段数
in = cat(1,in, zeros(fragNum*inFragLength-inputLength,2)); %%补零
outFragLength = round(bufferSize*rate); %%输出信号分段长度
outputLength = fragNum*outFragLength;   %%输出信号长度
out = zeros(outputLength,1);
realRate = outFragLength/inFragLength;  %%实际转换率
%%采样频率转换
for num = 1:fragNum
    inFrag = in(1+(num-1)*inFragLength:num*inFragLength,:);   %%分段操作
    inFragDCT = dct(inFrag);    %%离散余弦变换
    if  rate > 1
        %%插值操作/升采样
        outFragDCT = cat(1,inFragDCT,zeros(outFragLength-inFragLength,2));
        %%反离散余弦变换+幅度补偿
        outFrag = sqrt(rate)*idct(outFragDCT);
    else
        %%抽取操作/降采样
        outFragDCT = inFragDCT(1:outFragLength,:);
        %%反离散余弦变换+幅度补偿
        outFrag = sqrt(rate)*idct(outFragDCT);
    end
    out(1+(num-1)*outFragLength:num*outFragLength,1:2) = outFrag;
end
```

```
function pitch_exp(fileName)
fileName = 'I could be the one';
bufferSize = 2205;
RATE = [ 2^(2/12) 2^(1/12) 2^(0/12) 2^(-1/12) 2^(-2/12) ];
PITCH = cell(1,5);
PITCH{1} = '-2'; PITCH{2} = '-1'; PITCH{3} = '0'; PITCH{4} =
'+1'; PITCH{5} = '+2';
[ data fs  ] = audioread([ fileName '.wav' ]);
% for num = 1:size(RATE,2)
%     rate = RATE(num);
%     [ dataPitch realRate ] = src_dct(data, rate, bufferSize);
%     audiowrite([ fileName  PITCH{num} '.wav' ],dataPitch,
fs);
% end
[ dataPitch realRate ] = src_dct(data, 1.1, bufferSize);
audiowrite('new.wav' ,dataPitch./max(dataPitch), 48510);
```

3. 利用离散余弦变换的方式实现五阶音阶变换。

将改变采样率后的信号按原采样率播放即可

```
function pitch_exp(fileName)
fileName = 'I could be the one';
bufferSize = 2205;
RATE = [ 2^(2/12) 2^(1/12) 2^(0/12) 2^(-1/12) 2^(-2/12) ];
PITCH = cell(1,5);
PITCH{1} = '-2'; PITCH{2} = '-1'; PITCH{3} = '0'; PITCH{4} = '+1'; PITCH{5} = '+2';
[ data fs  ] = audioread([ fileName '.wav' ]);
 for num = 1:size(RATE,2)
    rate = RATE(num);
    [ dataPitch realRate ] = src_dct(data, rate, bufferSize);
    audiowrite([ fileName  PITCH{num} '.wav' ],dataPitch, fs);
 end
```

其中的 src_dct 函数与第二问相同

五、 实验数据记录和处理

1.利用升采样降采样的方式实现 44.1kHz 到 48kHz 的采样频率转换；

得到 10s 音频文件如下：

上方 oldData 是 44.1kHz，下方 out 是 48kHz。同样时间下数据大小具有相同的比例 441:480，说明频率转换成功。

2.利用离散余弦变换的方式实现 44.1kHz 到 48kHz 的采样频率转换

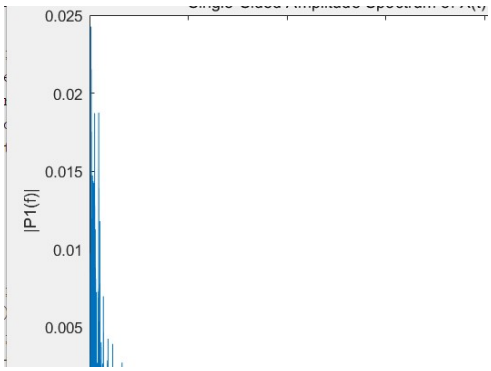
buffersize = 2205 时，经过升采样后，10s 的 44100 的音频数据变为 10s 的 48240 的音频数据

3.利用离散余弦变换的方式实现五阶音阶变换。

实现了音调变换，但发现音调变高，时间变短，音调变低，时间变长。

六、实验结果与分析

升采样降采样 SRC 能成功进行频率转换，但所得音质下降明显，并且仍存在刺耳的高频成分



使用离散余弦变换 SRC 后音质更加通透，没有明显的杂音，但改变的采样率有误差。
使用离散余弦变换 SRC 后按照原采样率播放，成功实现了音调变换，但是歌曲总体时间也改变了。

思考题

1． 为什么要实现采样频率转换？

从标准音频格式的角度来说，存在着很多的音频标准

格式：如 CD 格式，DAT 格式，MP3 格式等。由于这些格式具有不同的采样频率，所以要进行格式转换时必然需要进行采样频率转换。还有 intelAC’97 规范约定了声卡需要经过一个处理过程，即将所有信号重新转换成统一的采样率输出，因此也需要进行采样频率转换。

2.插值与滤波的关系？

将信号进行插值，然后进行重采样，即可完成滤波。

实验二 彩色数字图像及视频基础

一、 实验目的和要求

a) 熟练掌握 matlab 中常用的各种简单命令；
b) 了解常用的图像文件格式及 bmp 文件的结构；
c) 用 matlab 读取图像文件并显示出来
d) 用 matlab 对图像进行简单变换

二、 实验原理和内容

常用 bmp 文件结构

bmp 文件大体上分成四个部分，如表 1 所示。

表 1 Windows 位图文件结构示意图

位图文件头 BITMAPFILEHEADER
位图信息头 BITMAPINFOHEADER
调色板 Palette
实际的位图数据 ImageDate

第一部分为位图文件头**BITMAPFILEHEADER**，是一个结构，这个结构的长度是固定的，为 14个字节(WORD为无符号16位整数，DWORD为无符号32位整数)

第二部分为位图信息头 BITMAPINFOHEADER，也是一个结构，这个结构的长度是固定的，为 40个字节(LONG为 32位整数)

第三部分为调色板 Palette，当然，这里是对那些需要调色板的位图文件而言的。有些位图，如真彩色图，前面已经讲过，是不需要调色板的，BITMAPINFOHEADER 后直接是位图数据。

第四部分是实际的图像数据。对于用到调色板的位图，图像数据就是该像素颜在调色板中的索引值。对于真彩色图，图像数据就是实际的 R、G、B 值。

1． 调用 imread()函数读取图像并显示出来；
2． 调用 rgb2gray()把图像转化成灰度图像并显示出来；

3. 调用 imfinfo()可以查看 bmp 文件头信息。
4. 调用 imrotate()使图像旋转，并显示旋转后的图像
5. 调用 imresize()命令改变图像的大小

三、 主要仪器设备（软件环境）

Matlab R2019b

四、实验步骤与操作方法、实验数据记录和处理、实验结果与分析

1.调用 imread()函数读取图像并显示出来

调用 immread()函数读取图像并显示出来

imRGB=imread('C:\Users\LYM\Desktop\imrgb.jpg');

原图：

imshow(imRGB);　　figure(1),imshow(imRGB);

图 1 原图

2.调用 rgb2gray()把图像转化成灰度图像并显示出来；

灰度图：

```
imGray=rgb2gray(imRGB);

figure(2),imshow(imGray);
```

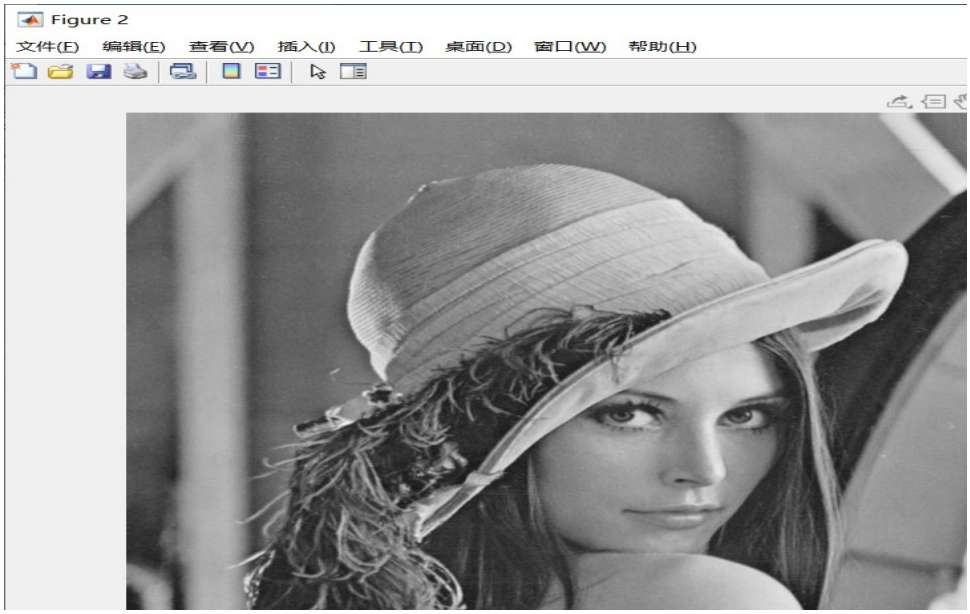


图 2 灰度图

```
whos imRGB
```

Name	Size	Bytes	Class	Attributes
imRGB	720x720x3	1555200	uint8	

```
whos imGray
```

Name	Size	Bytes	Class	Attributes
imGray	720x720	518400	uint8	

存图片为 bmp 格式：

```
imwrite(imRGB,'C:\Users\LYM\Desktop\lenabmp.bmp','bmp');
```

bmp：

图 3 bmp 格式图

3. 调用 imfinfo()可以查看 jpg 文件头信息。

包含以下字段的 struct:

```
            Filename: 'C:\Users\LYM\Desktop\imrgb.jpg'
        FileModDate: '07-Oct-2021 22:05:32'
            FileSize: 75126
              Format: 'jpg'
       FormatVersion: ''
               Width: 720
              Height: 720
            BitDepth: 24
           ColorType: 'truecolor'
     FormatSignature: ''
     NumberOfSamples: 3
      CodingMethod: 'Huffman'
     CodingProcess: 'Sequential'
             Comment: {}
```

调用 imfinfo()可以查看 bmp 文件头信息。

包含以下字段的 struct:

Filename: 'C:\Users\LYM\Desktop\lenabmp.bmp'

FileModDate: '10-Oct-2021 21:00:35'

FileSize: 1555254

Format: 'bmp'

FormatVersion: 'Version 3 (Microsoft Windows 3.x)'

Width: 720

Height: 720

BitDepth: 24

ColorType: 'truecolor'

FormatSignature: 'BM'

NumColormapEntries: 0

Colormap: []

RedMask: []

GreenMask: []

BlueMask: []

ImageDataOffset: 54

BitmapHeaderSize: 40

NumPlanes: 1

CompressionType: 'none'

BitmapSize: 1555200

HorzResolution: 0

VertResolution: 0

NumColorsUsed: 0

NumImportantColors: 0

4.调用 imrotate()使图像旋转，并显示旋转后的图像

旋转：

```
I = fitsread('solarspectra.fts');

I = mat2gray(I);

J = imrotate(I,-1,'bilinear','crop');

figure,imshow(I),figure,imshow(J);
```

```
I = fitsread('solarspectra.fts');

I = mat2gray(I);

J = imrotate(I,-60,'bilinear','crop');

figure,imshow(I),figure,imshow(J);
```

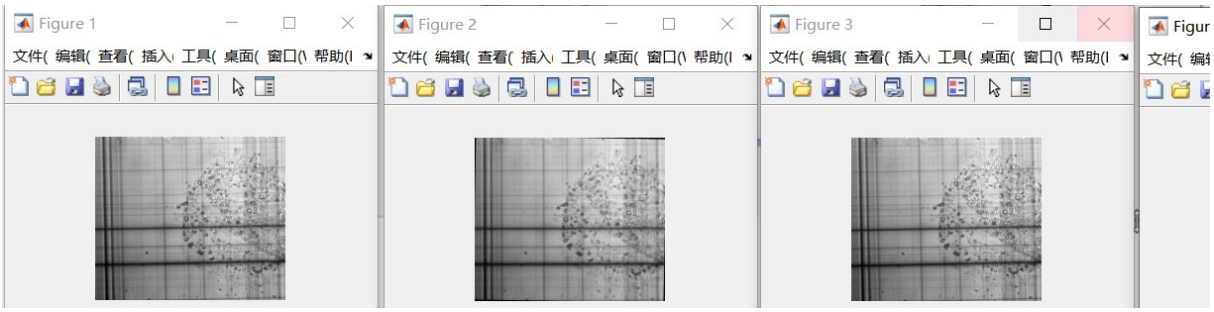


图 4

5.调用 imresize()命令改变图像的大小

```
I = imread('ngc6543a.jpg');

J = imresize(I, 0.5);
```

figure, imshow(I), figure, imshow(J)

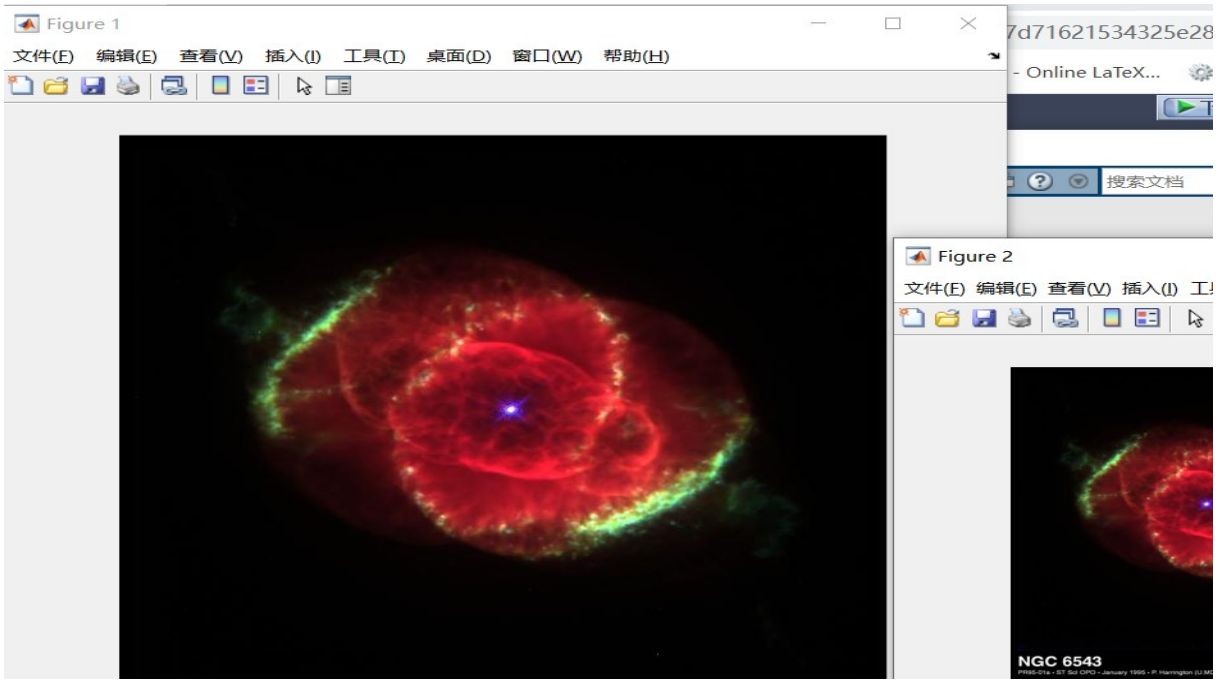


图 5　缩放为 0.5 倍

I = imread('rice.png');

J = imresize(I, 0.5);

figure, imshow(I), figure, imshow(J)

缩放后的图像与原图对比：

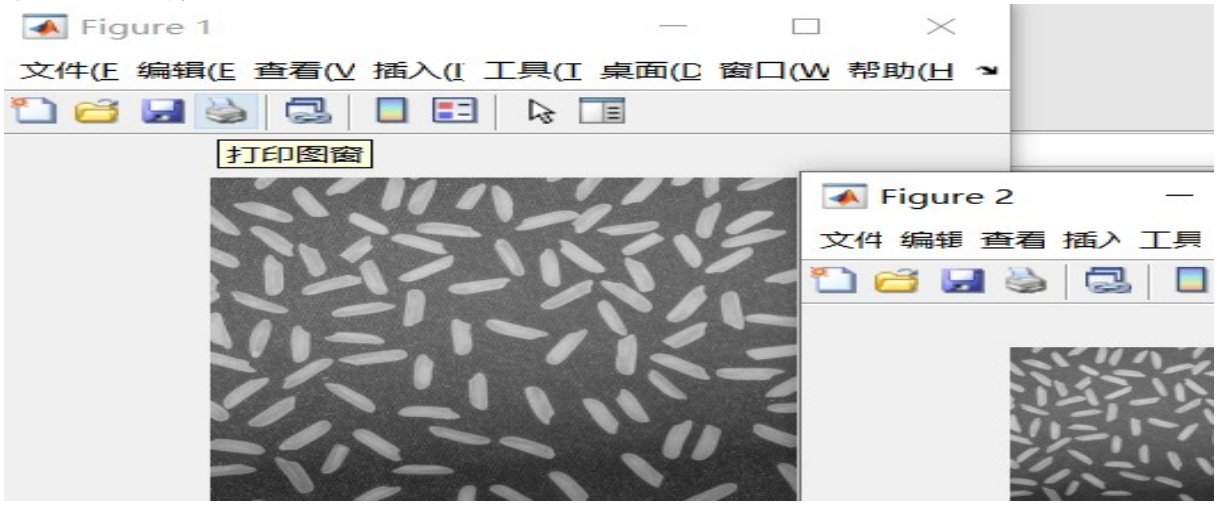


图 6 缩放 0.5 倍

四．思考题

1.程序中读取的是 jpg 图像，当读取的是 bmp 图像时 Infor 显示的什么？

答案：同实验内容 3

```
                Filename: 'C:\Users\LYM\Desktop\lenabmp.bmp'
            FileModDate: '10-Oct-2021 21:00:35'
                FileSize: 1555254
                  Format: 'bmp'
           FormatVersion: 'Version 3 (Microsoft Windows 3.x)'
                   Width: 720
                  Height: 720
                BitDepth: 24
               ColorType: 'truecolor'
         FormatSignature: 'BM'
      NumColormapEntries: 0
                Colormap: []
                 RedMask: []
               GreenMask: []
                BlueMask: []
         ImageDataOffset: 54
        BitmapHeaderSize: 40
              NumPlanes: 1
         CompressionType: 'none'
              BitmapSize: 1555200
           HorzResolution: 0
```

VertResolution: 0

NumColorsUsed: 0

NumImportantColors: 0

2. 如果读取的不是 24 位彩色图而是灰度图时，又会有什么效果？

读进来的数据维度变成1维了，而彩色图片维度是3维

图 7 数据维度变成 1 维，为灰度图

包含以下字段的 struct:

Filename: 'C:\Users\LYM\Desktop\lenabmp.jpg'

FileModDate: '21-Oct-2021 22:48:02'

FileSize: 62691

Format: 'jpg'

FormatVersion: ''

Width: 720

Height: 720

BitDepth: 8

ColorType: 'grayscale'

FormatSignature: ''

NumberOfSamples: 1

CodingMethod: 'Huffman'

CodingProcess: 'Sequential'

Comment: {}

实验三 图像压缩

一、 实验目的和要求

1．理解有损压缩和无损压缩的概念；

2．理解图像压缩的主要原则和目的；

3．了解几种常用的图像压缩编码方式。

4．利用 MATLAB 程序进行图像压缩。

二、实验原理和内容

1.图像压缩原理

图像压缩的理想标准是信息丢失最少，压缩比例最大。不损失图像质量的压缩称为无损压缩，无损压缩不可能达到很高的压缩比；损失图像质量的压缩称为有损压缩，高的压缩比是以牺牲图像质量为代价的。

编码压缩方法有许多种，从不同的角度出发有不同的分类方法，从信息论角度出发可分为两大类。

（1）冗余度压缩方法，也称无损压缩、信息保持编码或嫡编码。具体说就是解码图像和压缩编码前的图像严格相同，没有失真，从数学上讲是一种可逆运算。

（2）信息量压缩方法，也称有损压缩、失真度编码或烟压缩编码。也就是说解码图像和原始图像是有差别的，允许有一定的失真。

应用在多媒体中的图像压缩编码方法，从压缩编码算法原理上可以分为以下 3 类：

（1）无损压缩编码种类

（2）有损压缩编码种类

（3）混合编码。

2. 用 DCT 压缩图像的过程为：

(1) 首先将输入图像分解为 8×8 或 16×16 的块，然后对每个子块进行二维 DCT 变换。

(2)将变换后得到的量化的 DCT 系数进行编码和传送，形成压缩后的图像格式。

用 DCT 解压的过程为：

(1)对每个 8×8 或 16×16 块进行二维 DCT 反变换。

(2)将反变换的矩阵的块合成一个单一的图像。

余弦变换具有把高度相关数据能量集中的趋势，DCT 变换后矩阵的能量集中在矩阵的左上角，右下的大多数的 DCT 系数值非常接近于 0。对于通常的图像来说，舍弃这些接近于 0 的 DCT 的系数值，并不会对重构图像的画面质量带来显著的下降。所以，利用 DCT 变换进行图像压缩可以节约大量的存储空间。压缩应该在最合理地近似原图像的情况下使用最少的系数。使用系数的多少也决定了压缩比的大小。

二、 主要仪器设备（软件环境）

Matlab R2019b

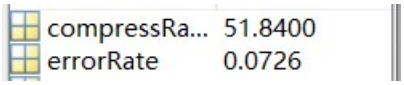
三、 实验步骤与操作方法

1. 利用 DCT 变换对图像进行 dct 变换，对比原始图像和 dct 反变换后图像的差异

```
function demo_dct

imRGB  = imread('imrgb.jpg');
imGray = double(rgb2gray(imRGB));

figure(1),imshow(imGray, [0 255]);%灰度图

imDct  = dct2(imGray);

figure(2),imshow(abs(imDct), [0 255]);@DCT变换

imRec  = idct2(imDct);

figure(3),imshow(imRec,[0 255]);%IDCT反变换

%
sz = [100, 100];

imDctN = imDct(1:sz(1),1:sz(2));%取前100维

imRecN = idct2(imDctN, size(imGray));%IDCT反变换

figure(4),imshow(imRecN, [0 255]);
errorRate = norm(imRecN(:)-imGray(:))/norm(imGray(:))
compressRate =
(size(imGray,1)*size(imGray,2))/(sz(1)*sz(2))
```

2.利用离散余弦变换进行 JPEG 图像压缩，并计算压缩比

压缩比计算



如图所示：压缩比为 51.8400

3.利用行程编码（RLE）对 checkerboard 图像进行图像压缩，并计算压缩比

```
data0= checkerboard(400);
data1=data0(:);

l_data1=length(data1); %计算长度

data2=im2bw(data0,0.5); %将原图转换为二值图像，阈值为 0.5

X=data2(:); %令 X为新建的二值图像的一维数据组

L=length(X);
j=1;
data3(1)=1;
```

```
for z=1:1:(length(X)-1) %行程编码程序段
if X(z)==X(z+1)
data3(j)=data3(j)+1;
else
end
data(j)=X(z); % data(j)代表相应的像素数据
j=j+1;
data3(j)=1;
end
data(j)=X(length(X)); %最后一个像素数据赋给 data

l_data3=length(data3); %计算行程编码后的所占字节数长度
l=1;
for m=1:l_data3
for n=1:1:l_data3(m);
decode_image1(l)=data(m);
l=l+1;
end
end
decode_image=reshape(decode_image1,[3200,3200]); %重建二维图像数组
h=figure;
subplot(131); imshow(data0);
title('原始图像'); %显示原图的二值图像
subplot(132); imshow(data2);
title('原图的二值图像'); %显示原图的二值图像
subplot(133);imshow(decode_image);
title('解压恢复后的图像'); %显示解压恢复后的图像

disp('原图像数据的长度：')
disp(L);
disp('压缩后图像数据的长度：')
disp(l_data3);
disp('解压后图像数据的长度：')
disp(length(decode_image1));
```

五、实验数据记录和处理

1. 对图像进行 DCT 变换，对比原始图像和 dct 反变换后图像的差异

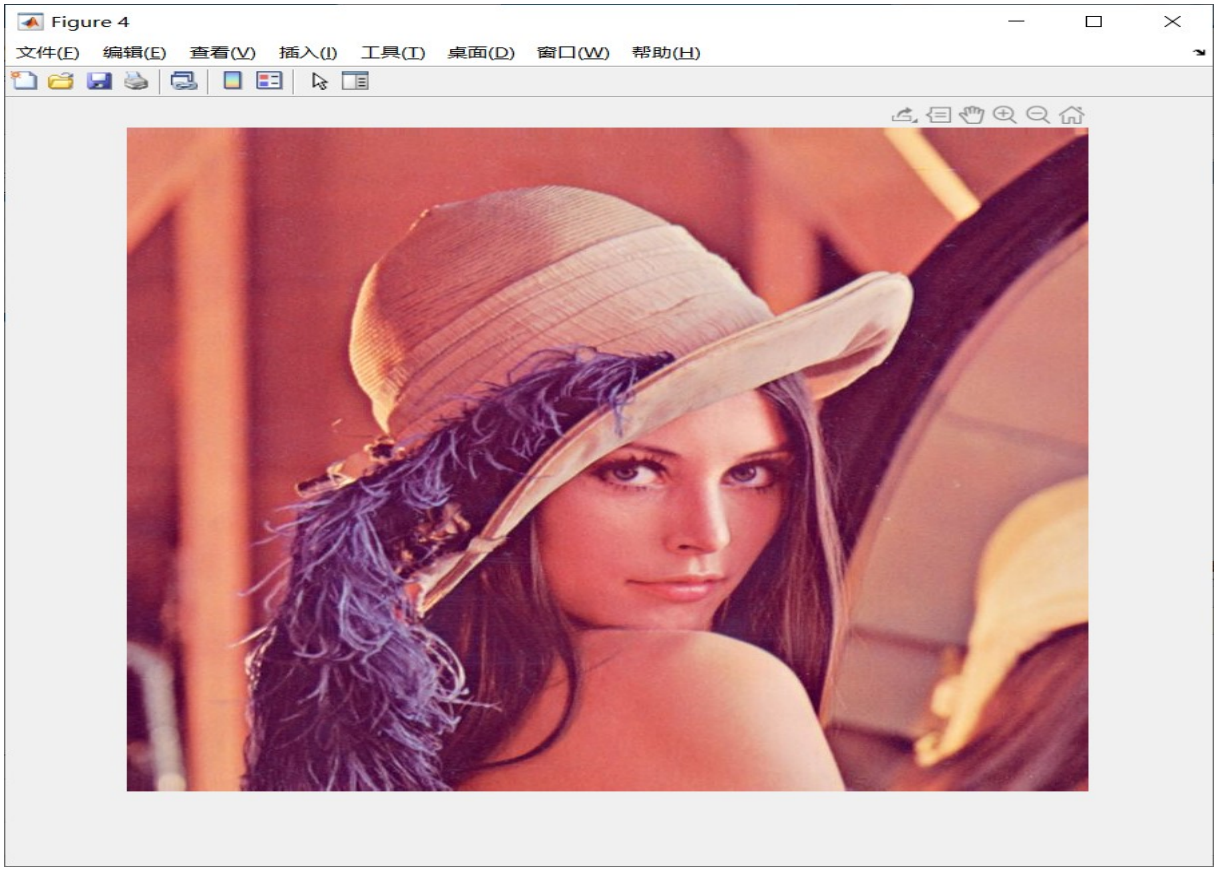


图 1 原图

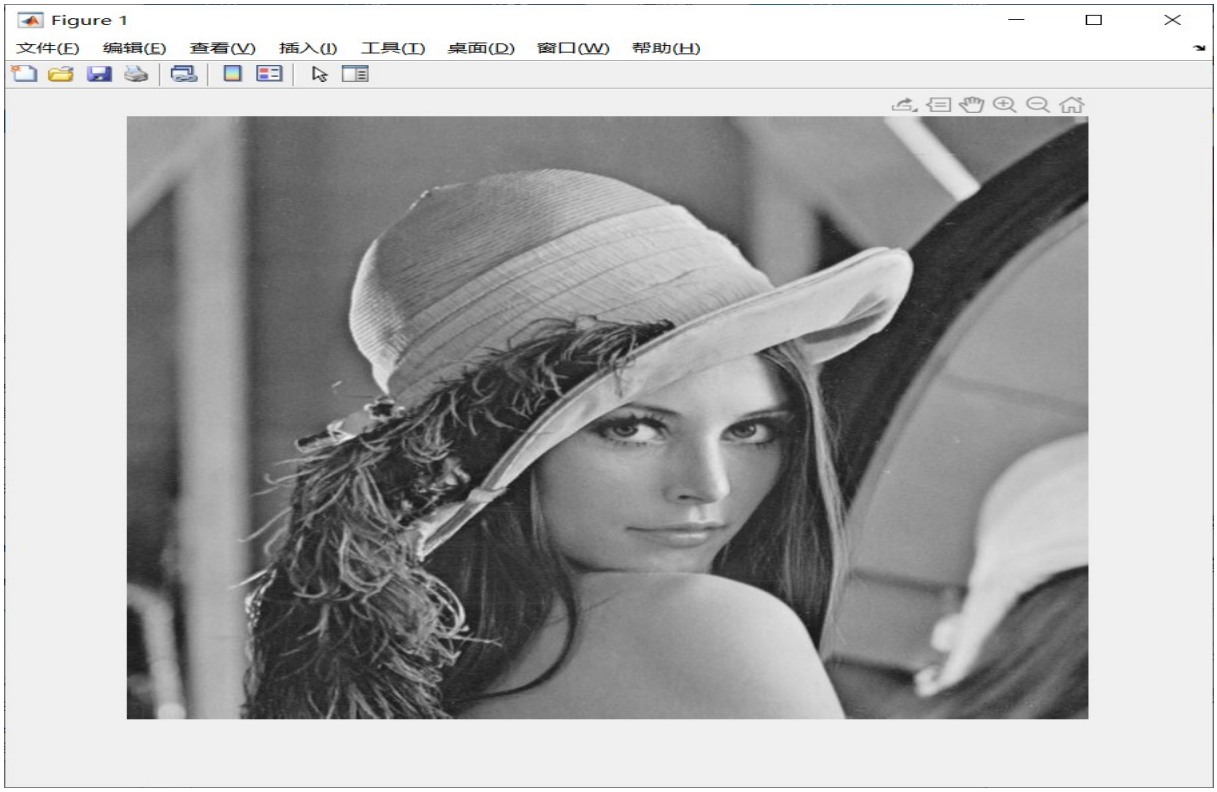


图 2 灰度图

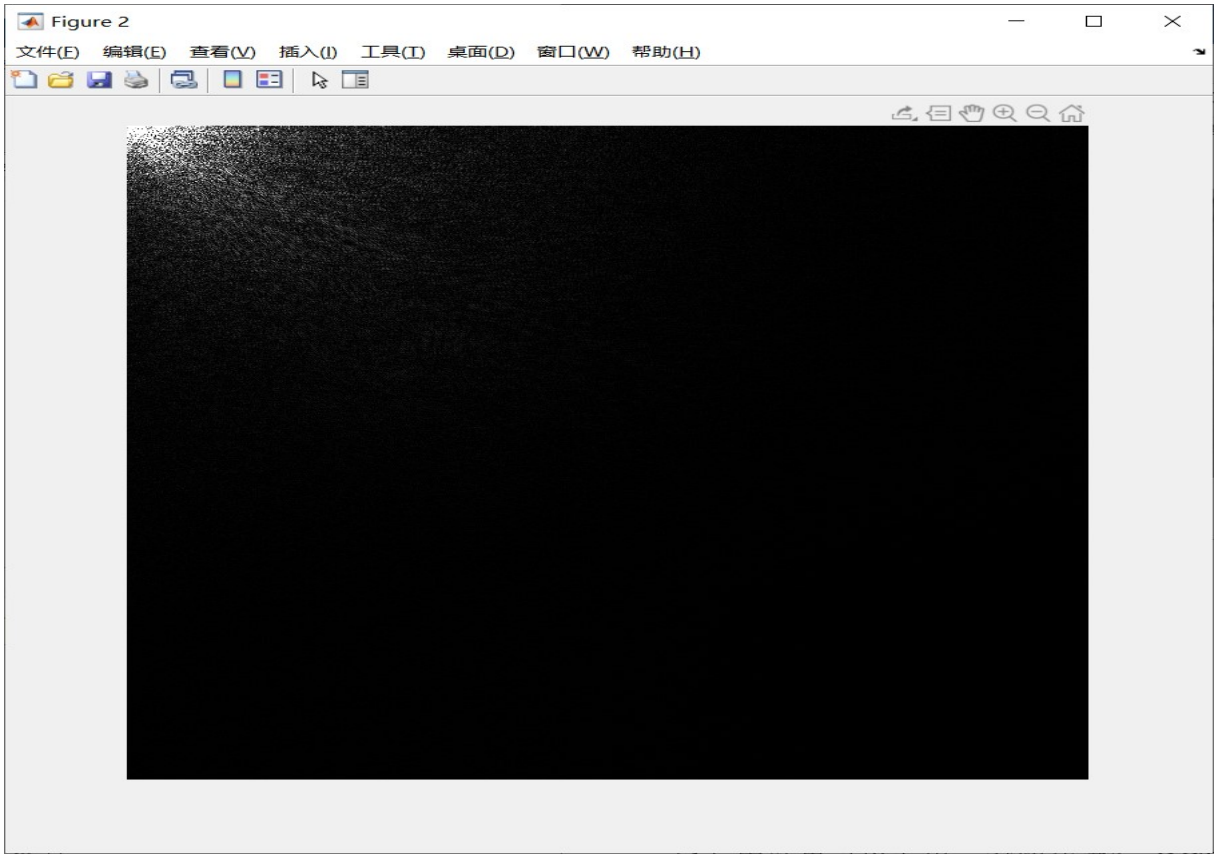


图 3 DCT 变换后的图片

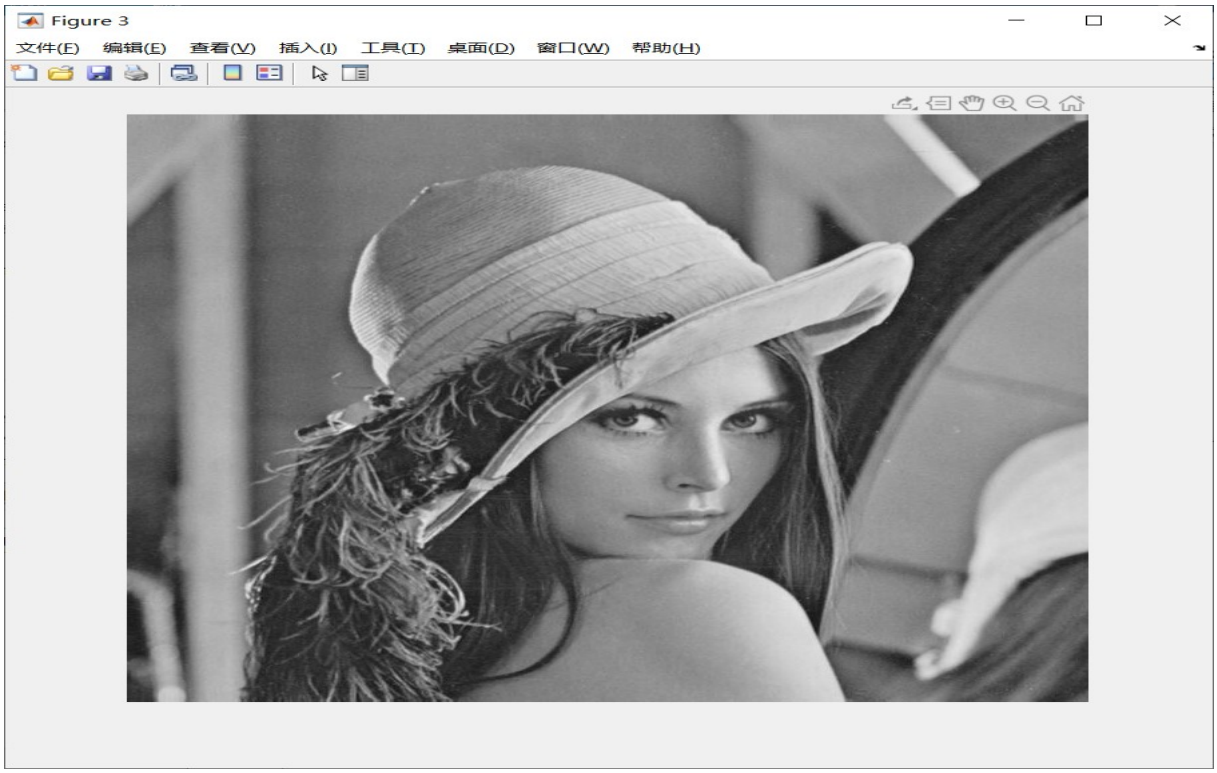


图 4 IDCT 反变换（未压缩）

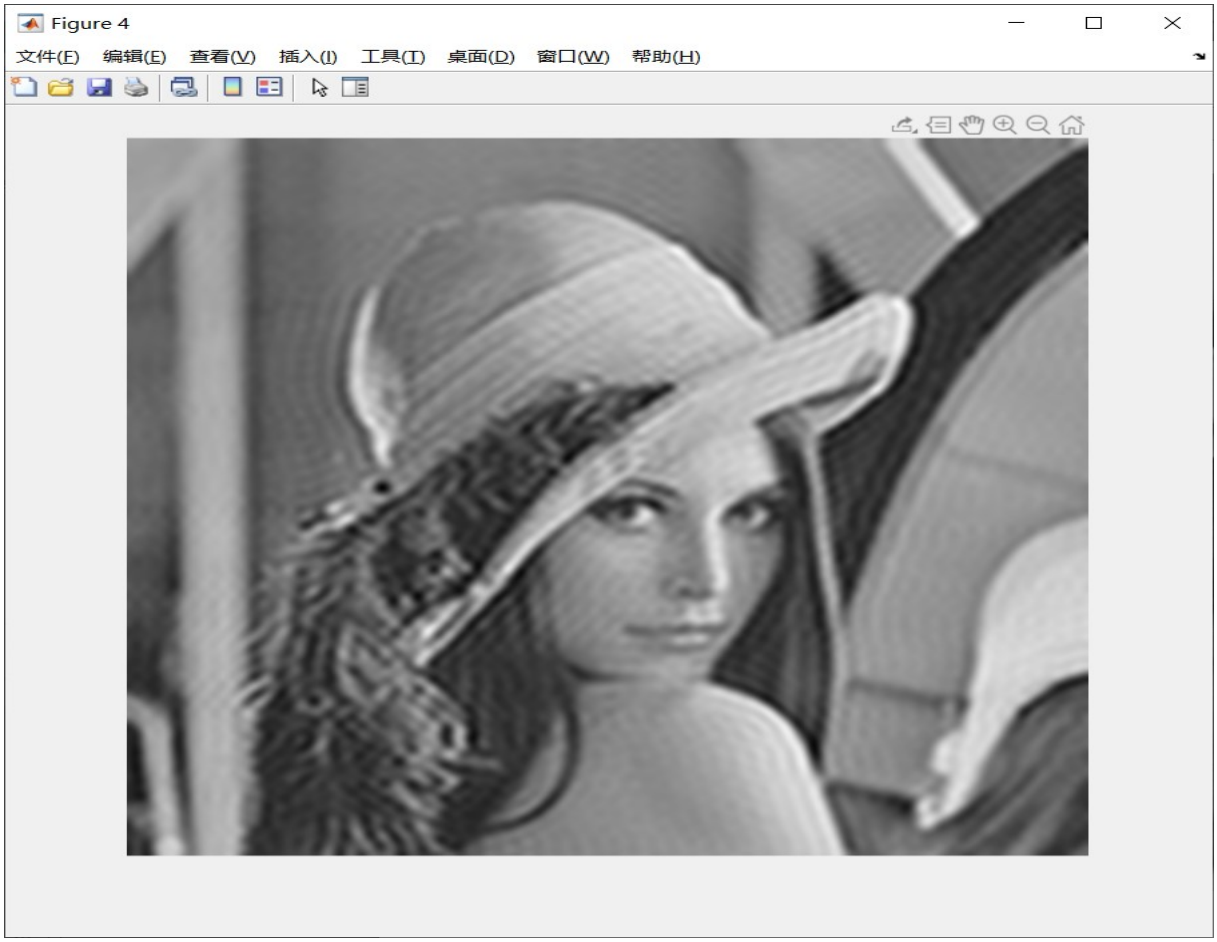


图 5 IDCT 反变换（压缩）

2. 利用行程编码（RLE）对 checkerboard 图像进行图像压缩，并计算压缩比

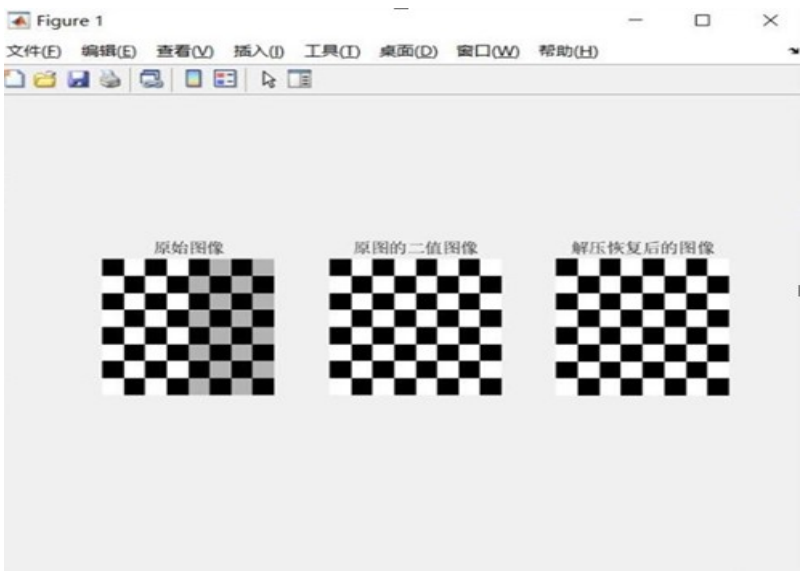


图 1　使用 RLE 对棋盘格图进行压缩

六、实验结果与分析

（1）由结果可以看出，对图片进行 DCT 变换后，高频点集中在左上角，反变换后的图片于原图相比差距不大；

（2）结果可知，压缩后的图片相比于原图清晰度下降；

（3）结果可知，对于灰度图和二值图，行程编码的压缩率达到了 400:1，有很高的压缩率。

七、讨论、建议、质疑

1．图像中哪些信息是主要的，哪些信息是次要的？

DCT 变换后图像变成 1、0，代号 0 的就是次要冗余信息,1 就是主要信息。

2、简述离散余弦变换（DCT）和行程编码（RLE）的原理

DCT:余弦变换具有把高度相关数据能量集中的趋势，DCT 变换后矩阵的能量集中在矩阵的左上角，右下的大多数的 DCT 系数值非常接近于 0。对于通常的图像来说，舍弃这些接近于 0 的 DCT 的系数值，并不会对重构图像的画面质量带来显著的下降。所以，利用 DCT 变换进行图像压缩可以节约大量的存储空间。RLE：RLE 算法的原理就是用一个表示块数的属性加上一个数据块代表原来连续的若干块数据 ,从而达到节省存储空间的目的。

PCA：

代码：

```
clc;
clear all;
%readPath = ['./Yale_64x64_bmp/'];                    %%原始图像
```

```
读取路径
readPath  =  ['./orl_48x48/'];                                    %%原始图像
读取路径
imNum      = 400;
height    = 48;
width      = 48;
data        = zeros(height*width, imNum);
dirPath = dir([readPath '*.bmp']);
for num = 1:imNum
        imGray            = imread([readPath dirPath(num).name]);
        %data(:,num) = matrix_to_vector(imGray(:,:,1));
        data(:,num) = reshape(imGray(:,:,1),height*width,1);
end
%%    PCA 算法
[U, mu] = pca_dw(data, 30, 0);
%%    显示特征脸（montage)
figure;
montage(uint8(reshape(mu,48,48)));
%%    特征提取
fea = U'*(data-repmat(mu,[1,size(data,2)]));
%%    特征重构
rec = U*fea + repmat(mu,[1,size(data,2)]);
```

```
%%   对比重构图和原图
figure;
montage(imGray(:,:,1));
figure;
bb=reshape(rec(:,400),48,48);
montage(uint8(bb));
%%   重构误差
rError = sum(sum((data-rec).^2))/sum(sum((data).^2));
%
cRate =
(size(fea,1)*size(fea,2)+size(mu,1)*size(mu,2)+size(U,1)*size(U,2))/(size(data,
1)*size(data,2));

figure;
[U, mu, eRate] = pca_dw(data, size(data,2), 0);
for dim = 1:size(data,2)
    fea = U(:,1:dim)'*(data-repmat(mu,[1,size(data,2)]));
    rec = U(:,1:dim)*fea + repmat(mu,[1,size(data,2)]);
    rError = sum(sum((data-rec).^2))/sum(sum((data).^2));
    cRate =
(size(fea,1)*size(fea,2)+size(mu,1)*size(mu,2)+size(U(:,1:dim),1)*size(U(:,1:di
m),2))/(size(data,1)*size(data,2));
```

```
    plot(dim, rError, '--rs');
    hold on;
    plot(dim, cRate,    '--b*');
    hold on;
    plot(dim, eRate(dim),    '--mo');
    hold on;
end


function [ vector , longth ] = matrix_to_vector( matrix )

%%function [ vector , longth ] = matrix_to_vector( matrix )
%%矩阵转化为一个列向量;
%%输入参数:
%%            matrix    :    一个矩阵;
%%输出参数:
%%            vector    :    转化后的向量;
%%            longth     :        向量的长度;
%%

matrix_size = size(matrix);
height = matrix_size(1);
```

```
width    = matrix_size(2);


longth = height * width;


temp = zeros(height*width,1);
for cl = 0:width-1
      temp(height*cl+1 : height*cl+height) = matrix(:,cl+1);
end
vector = temp;



function [U, mu, eRate] = pca_dw(data, dim, rate)
%%function [U, mu] = pca_dw(data, dim, rate)
%%Calculate PCA to obtain basis functions
%%Input:
%%          data:     data matrix [nDim*nSample]
%%          dim:       Reserved Dimensions
%%          rate:     Reserved Energy Rate, only work when 0 < rate < = 1
%%Output:
%%          mu:      mean vector [nDim*1]
%%          U:       basis functions [nDim*dim]
%%          eRate:    Reserved Energy Rate
```

```
[nDim, nSample] = size(data);
%
% % %%Calculate the mean
% % muT = zeros(nDim,1);
% % for num = 1:nSample
% %         muT = muT + data(:,num);
% % end
% % muT = muT / nSample;
%
%%Calculate the mean
mu      = mean(data, 2);
% % %%Remove the mean
% % dataT = data;
% % for num = 1:nSample
% %         dataT(:,num) = dataT(:,num) - mu;
% % end
%
%%Remove the mean
data = data - repmat(mu, [1, nSample]);
%
covM = data*data'/nSample;
```

```
    [U, D] = eig(covM);
    [val, idx] = sort(diag(D), 'descend');
    U = U(:,idx);
    %
    valsum = cumsum(val)./sum(val);
    if    rate~=0
        dim       = find((valsum>=rate), 1, 'first');
    end
    eRate    = valsum(1:dim);
    %
    U = U(:, 1:dim);
end
```

结果与分析：

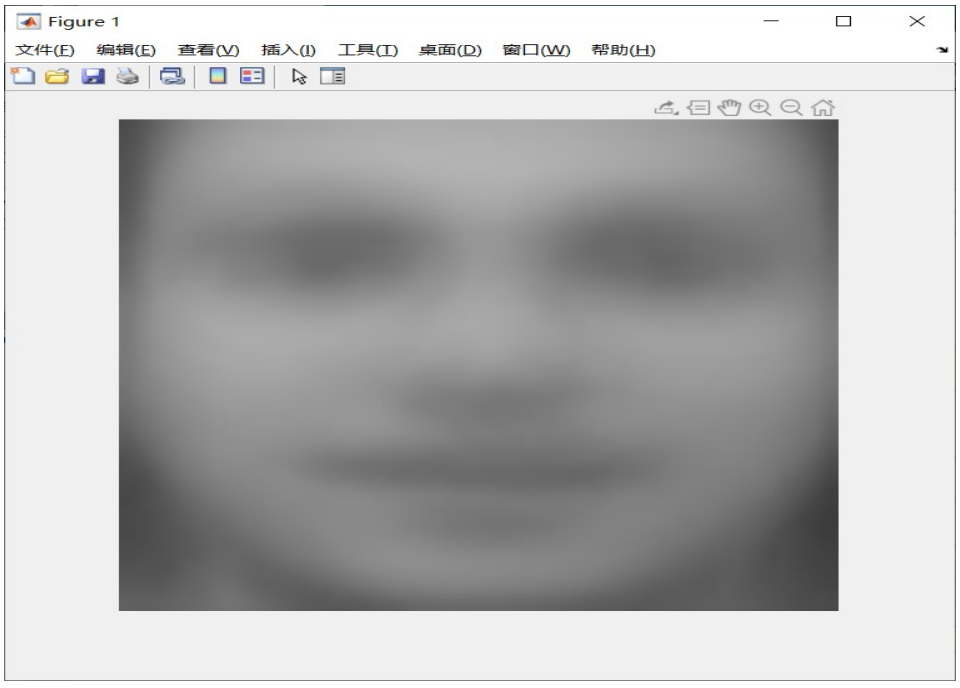


图 1 平均脸

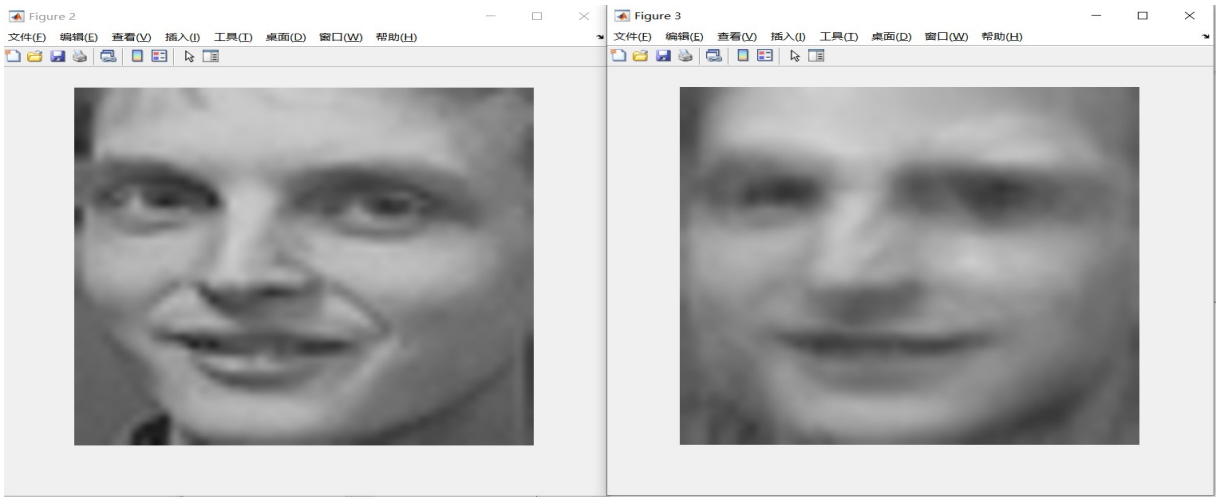


图 2 左原图，右压缩重构图

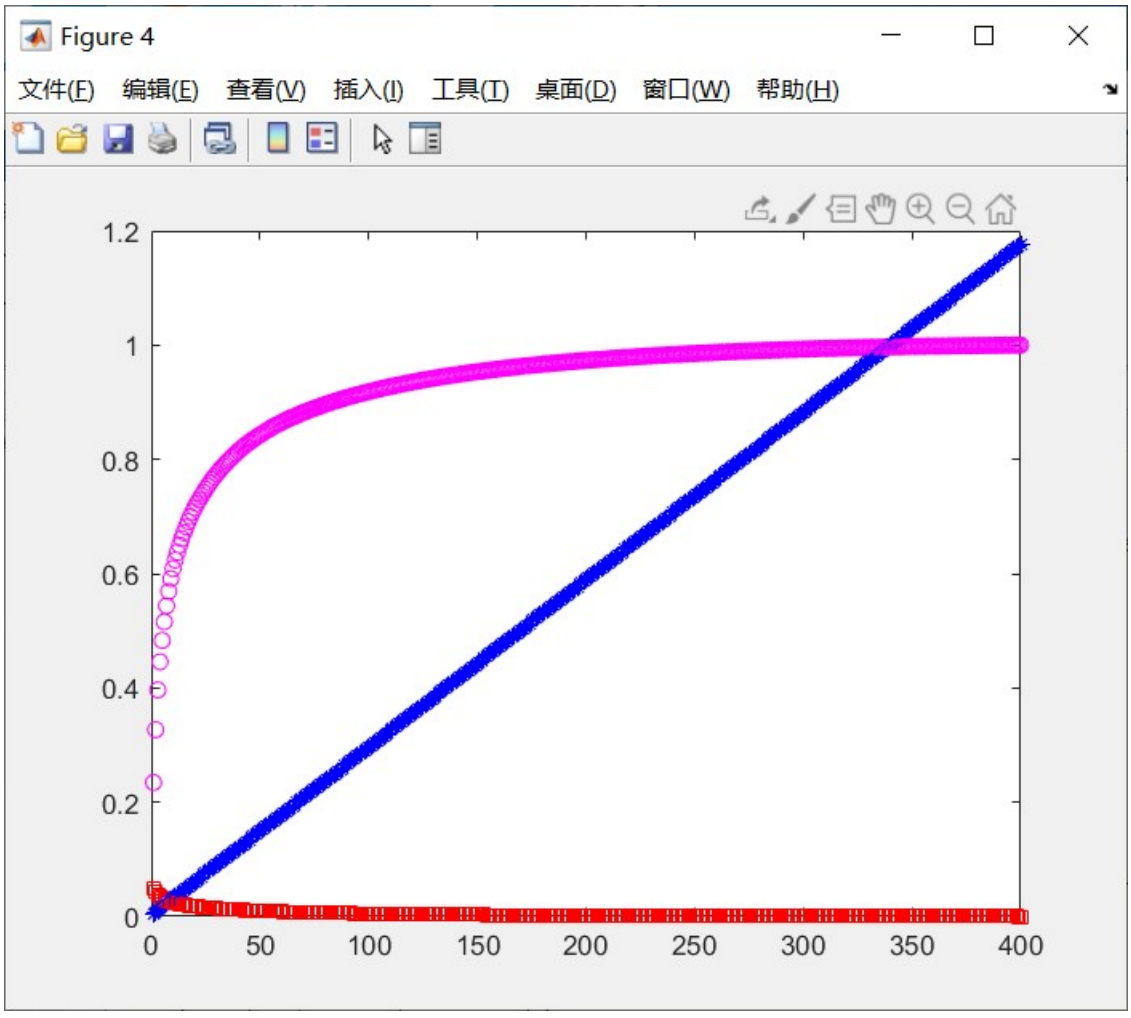


图3 横轴维度，纵轴红色误差，蓝色压缩率，锰紫色为eRate(dim)