图像处理大作业

一、基础知识

练习题：

1.MATLAB 提供了图像处理的工具箱，在命令窗口输入 help images 可查看该工具箱内的所有函数，请阅读并大致了解这些函数的基本功能。

命令行窗口

```
>> help images
images 的内容:

getrangefromclass            - Get dynamic range of image based on its class.
hsv2rgb                      - Convert hue-saturation-value colors to red-green-blue.
im2double                    - Convert image to double precision.
imshow                       - Display image in Handle Graphics figure.
isdicom                      - Determine if a file is probably a DICOM file.
isdpx                        - Check if file is DPX.
isnitf                       - Check if file is NITF.
rgb2gray                     - Convert RGB image or colormap to grayscale.
rgb2hsv                      - Convert red-green-blue colors to hue-saturation-value.


  Image Processing Toolbox
  Version 9.4 (R2016a) 10-Feb-2016

  Image display, exploration, and visualization.
    colorbar       - Display colorbar (MATLAB Toolbox).
    image          - Create and display image object (MATLAB Toolbox).
    imagesc        - Scale data and display as image (MATLAB Toolbox).
    immovie        - Make movie from multiframe image.
    imoverlay      - Burn binary mask into a 2-D image.
    implay         - Play movies, videos, or image sequences.
    imshow         - Display image in Handle Graphics figure.
    imtool         - Display image in the Image Tool.
    montage        - Display multiple image frames as rectangular montage.
    movie          - Play recorded movie frames (MATLAB Toolbox).
    subimage       - Display multiple images in single figure.
    visboundaries  - Plot region boundaries.
    viscircles     - Create circle.
    warp           - Display image as texture-mapped surface.

  Image file I/O.
    analyze75info       - Read metadata from header file of Mayo Analyze 7.5 data set.
    analyze75read       - Read image file of Mayo Analyze 7.5 data set.
    dicomanon           - Anonymize DICOM file.
    dicomdict           - Get or set active DICOM data dictionary.
    dicominfo           - Read metadata from DICOM message.
    dicomlookup         - Find attribute in DICOM data dictionary.
    dicomread           - Read DICOM image.
```

通过 help images 命令可以看出有 MATLAB 提供了大量的图形处理相关的函数，通过对其的阅读，已经大致了解了这些函数的用法及基本功能。

2. 利用 MATLAB 提供的 Image file I/O 函数分别完成以下处理：

（a）以测试图像的中心为圆心，图像的长和宽中较小值的一半为半径画一个红颜色的圆；

解题思路：由于实验中给出的 hall.mat 文件中的 hall_color 的 size 为 120*168*3，所以需要找出测试图像的中点，然后遍历所有像素点，如果到中点的距离小于等于 60（取长或宽中较小值的一半），则将其决定颜色的数组置为[255,0,0],当遍历完所有的像素点后，即可在原测试图像中画出以图像的长和宽中较小值的一半为半径的红颜色的圆。

完整代码保存在 solved_a.m 中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
[leng,wide,rbg]=size(hall_color);
circlecenter = [leng/2,wide/2];                    %找出圆心
max_r=min(circlecenter(1),circlecenter(2));        %确定半径范围
hall_a = hall_color;
for i = 1:leng                                     %遍历
    for j = 1:wide
        r=sqrt((i-circlecenter(1))^2 + (j-circlecenter(2))^2); %计算到圆心的距离
        if(r<=max_r)
            hall_a(i,j,:)=[255,0,0];               %在最大半径范围内，将颜色变为红色
        end
    end
end
imshow(hall_a);
imwrite(hall_a,'hall_a.jpg');
```

得到的图像如下图所示：

（b）将测试图像涂成国际象棋状的“黑白格”的样子，其中“黑”即黑色，“白”则意味着保留原图。用一种看图软件浏览上述的两个图，看是否达到了目标。

解题思路：本问与上一问的思路大致相同，同样需要对测试图像的像素点进行遍历，不过判断条件有所改变，根据国际象棋是 8*8 的布局，如果将横向与纵向都用 1:8 进行坐标编号，则棋盘具有以下规律：横纵坐标之和为奇数的方格为“白”，为偶数的为“黑”。将此规律作为判断依据即可。

完整代码保存在文件 solved_b.m 中：

```
close all,clear all,clc;
load('hall.mat');
[leng,wide,rbg]=size(hall_color);
hall_b = hall_color;
for i = 1:8                                    %遍历
    for j = 1:8
        if(mod(i+j,2)==0)
        %若所在方格为黑，则将方格内的所有点置为黑色
        for m=(i-1)*(leng/8)+1:i*(leng/8)
            for n=(j-1)*(wide/8)+1:j*(wide/8)
                hall_b(m,n,:)=[0,0,0];
            end
        end
        end
    end
end
imshow(hall_b);
imwrite(hall_b,'hall_b.jpg');
```

得到的图像如下图所示：

由结果截图可知，均达到题目要求。

二 、图像压缩编码

练习题：

1.图像的预处理是将每个像素灰度值减去 128，这个步骤是否可以在变换域进行？请在测试图像中截取一块验证你的结论。

解题思路：任取一个 10*10 的矩阵进行观察，最简单的操作就是直接将选取的矩阵-128，如果在变换域进行的话，需要将选取的矩阵先做 DCT 变换，然后减去直流项，即 128*10，然后通过 idct 还原即可。

代码已经保存到 solved1.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
hall_part=hall_gray(11:20,11:20);
hall_1=hall_part-128;
temp_dct = dct2(hall_part);
temp_dct(1) = temp_dct(1) -128/(1/10);
hall_2=idct2(temp_dct);
```

运行结果如下：

```
hall_1 =

  107  104  102  103  109  113  110  110  110  111
  105  105   98  101  108  108  109  110  110  110
  100   97   93   98  104  108  107  108  105  108
  101   94   89   94   99  104  106  106  106  103
   98   97   91   90   94  100   99   99   98   97
   93   92   89   90   89   92   94   89   92   92
   83   85   86   87   85   86   86   86   86   84
   76   76   80   81   80   78   80   83   80   81
   66   69   71   66   64   69   69   71   78   77
   64   67   65   62   65   64   66   72   77   77

>> hall_2

hall_2 =

  107.0000  104.0000  102.0000  103.0000  109.0000  113.0000  110.0000  110.0000  110.0000  111.0000
  105.0000  105.0000   98.0000  101.0000  108.0000  108.0000  109.0000  110.0000  110.0000  110.0000
  100.0000   97.0000   93.0000   98.0000  104.0000  108.0000  107.0000  108.0000  105.0000  108.0000
  101.0000   94.0000   89.0000   94.0000   99.0000  104.0000  106.0000  106.0000  106.0000  103.0000
   98.0000   97.0000   91.0000   90.0000   94.0000  100.0000   99.0000   99.0000   98.0000   97.0000
   93.0000   92.0000   89.0000   90.0000   89.0000   92.0000   94.0000   89.0000   92.0000   92.0000
   83.0000   85.0000   86.0000   87.0000   85.0000   86.0000   86.0000   86.0000   86.0000   84.0000
   76.0000   76.0000   80.0000   81.0000   80.0000   78.0000   80.0000   83.0000   80.0000   81.0000
   66.0000   69.0000   71.0000   66.0000   64.0000   69.0000   69.0000   71.0000   78.0000   77.0000
   64.0000   67.0000   65.0000   62.0000   65.0000   64.0000   66.0000   72.0000   77.0000   77.0000
```

可以看出，二者得到的结果完全一样，所以可以在变换域进行。

2.请编程实现二维 DCT，并和 MATLAB 自带的库函数 dct2 比较是否一致。

解题思路：根据实验指导书中的二维 DCT 的计算公式，直接实现即可，比较简单。

完整代码保存在 solved2.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
hall_part = hall_gray(11:18,21:28);
N = 8;
[M,N]=size(hall_part);
D = zeros(M,N);
for i = 1:N
    for j = 1:N
        if(i==1)
           D(i,j) = sqrt(1/N)*cos(pi*(i-1)*(2*j-1)/(2*N));  %根据实验指导书原理D的第一行系数对应的ai=sqrt（1/N）
        else
           D(i,j) = sqrt(2/N)*cos(pi*(i-1)*(2*j-1)/(2*N));
        end
    end
end
P = double(hall_part);
myDCT2 = D*P*D';
DCT2 = dct2(hall_part);
```

截取部分图像作二维 DCT 变换，对比结果如下：

命令行窗口

```
>> myDCT2

myDCT2 =

   1.0e+03 *

    1.6743    0.0828    0.0022   -0.0043    0.0045    0.0007    0.0026   -0.0011
    0.0587    0.0270   -0.0099    0.0158   -0.0112    0.0060   -0.0031    0.0027
   -0.0288    0.0137   -0.0151    0.0085   -0.0015    0.0023    0.0008    0.0011
    0.0029   -0.0025    0.0006    0.0027   -0.0013   -0.0000   -0.0004   -0.0004
   -0.0025   -0.0013    0.0010    0.0034   -0.0012   -0.0004   -0.0009    0.0007
    0.0012    0.0006   -0.0008    0.0004    0.0003    0.0000    0.0021   -0.0000
   -0.0003    0.0001   -0.0012    0.0015   -0.0004   -0.0000    0.0001    0.0014
    0.0013    0.0002    0.0015    0.0000   -0.0009    0.0011   -0.0001   -0.0017

>> DCT2

DCT2 =

   1.0e+03 *

    1.6742    0.0828    0.0022   -0.0043    0.0045    0.0007    0.0026   -0.0011
    0.0587    0.0270   -0.0099    0.0158   -0.0112    0.0060   -0.0031    0.0027
   -0.0288    0.0137   -0.0151    0.0085   -0.0015    0.0023    0.0008    0.0011
    0.0029   -0.0025    0.0006    0.0027   -0.0013   -0.0000   -0.0004   -0.0004
   -0.0025   -0.0013    0.0010    0.0034   -0.0013   -0.0004   -0.0009    0.0007
    0.0012    0.0006   -0.0008    0.0004    0.0003    0.0000    0.0021   -0.0000
   -0.0003    0.0001   -0.0012    0.0015   -0.0004   -0.0000    0.0001    0.0014
    0.0013    0.0002    0.0015    0.0000   -0.0009    0.0011   -0.0001   -0.0017
```

可以看出，自己实现的二维 DCT 变换和 MATLAB 提供的函数计算结果基本一致，只有在直流分量上有微小差异，经过多组数据反复试验，发现均只在直流分量相差 0.0001，其余部分完全相等，故可认为自己编程实现的二维 DCT 变换功能正确。

3. 如果将 DCT 系统矩阵中右侧四列的系数全部置零，逆变换后的图像会发生什么变化？选取一块图验证你的结论。如果左侧四列全部置零呢？

解题思路：先直接将对应图片做 DCT 变换，之后分别将 DCT 系数矩阵的左边 4 列和右边 4 列置为 0 即可，然后用 idct2 函数反变换回来即可。需要注意的是，由于 IDCT 得到的结果是 double 型，在使用 imsho 和 imwrite 函数时需要映射到[0，1]区间，可以处以 256 实现。

完整代码保存在 solved3.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
hall_part=hall_gray;
hall_dct=dct2(hall_part);
[leng,wide,rpg]=size(hall_dct);
dctr0(:,1:wide-4)=hall_dct(:,1:wide-4);
dctr0(:,wide-3:wide)=0;
dctl0(:,5:wide)=hall_dct(:,5:wide);
dctl0(:,1:4)=0;
rightzero=idct2(dctr0)/256; %由于idct变换的结果为double型，需要归一化
leftzero=idct2(dctl0)/256;
subplot(1,3,1);imshow(hall_gray);title('原图');
subplot(1,3,2);imshow(rightzero);title('右边4列置0');
subplot(1,3,3);imshow(leftzero);title('左边4列置0');
imwrite(hall_gray,'hall_gray.jpg');
imwrite(rightzero,'rightzero.jpg');
imwrite(leftzero,'leftzero.jpg');
```

得到的结果如下：

上图从左往右依次是原图，右边 4 列为 0，左边 4 列为 0。可以看出右边 4 列为 0 时只有些许模糊，变化不大，但是左边 4 列置为 0 后，图像严重失真变暗。结合实验指导书的相关背景知识可知，右边 4 列置为 0 消除的是高频分量，所以对图像影响相对较小，而左边 4 列置为 0 则是消除了低频分量，对图像的影响将非常大。

4.若对 DCT 系数分别做转置，旋转 90 度和旋转 180 度操作（rot90），逆变换后恢复的图像有何变化？选取一块图验证你的结论。

解题思路：先将选取的图片区域做二维 DCT 变换，之后再将 DCT 系数矩阵分别做转置，旋转变换，最后将得到的结果做 IDCT 变换即可，同样需要注意归一化。

完整代码保存在 solved4.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
hall_part=hall_gray(1:100,1:100);
hall_dct=dct2(hall_part);
[leng,wide]=size(hall_dct);
hall_rt=hall_dct';
hall_rot90=rot90(hall_dct);
hall_rot180=rot90(rot90(hall_dct));
RT=idct2(hall_rt)/256;      %由于idct变换的结果为double型，需要归一化
ROT90=idct2(hall_rot90)/256;
ROT180=idct2(hall_rot180)/256;
subplot(2,2,1);imshow(hall_part);title('原图区域');
subplot(2,2,2);imshow(RT);title('转置');
subplot(2,2,3);imshow(ROT90);title('旋转90度');
subplot(2,2,4);imshow(ROT180);title('旋转180度');
imwrite(RT,'RT.jpg');
imwrite(ROT90,'ROT90.jpg');
imwrite(ROT180,'ROT180.jpg');
```

结果如下图所示：

选取原图了左上角 100*100 的区域分别做转置，旋转 90，旋转 180，得到的结果如上图所示。可以看出转置后相当于将原图先做镜面变换，再旋转，图片质量没多大影响。而旋转 90 和旋转 180 后，可以看出图片明显失真。

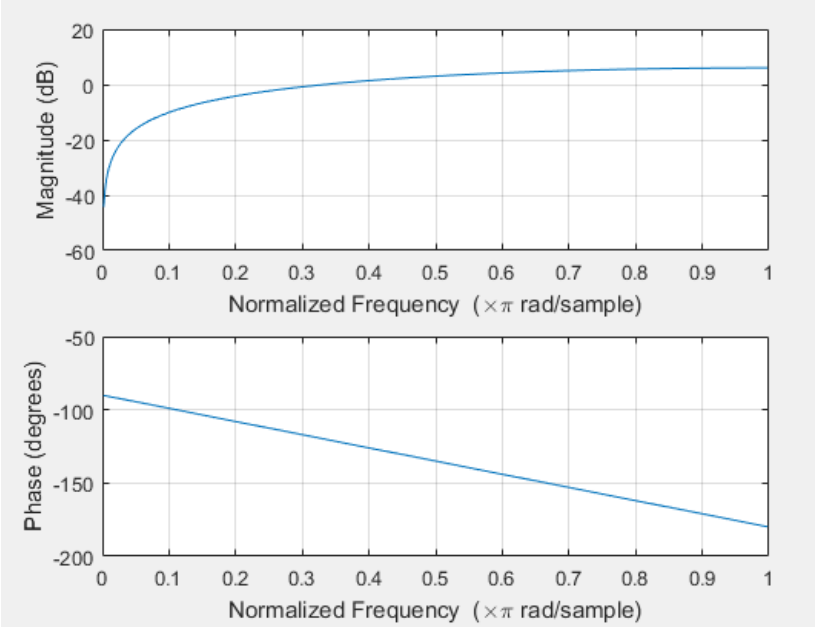
5. 如果认为差分编码是一个系统，请绘出这个系统的频率响应，说明它是一个_____（低通、高通、带通、带阻）滤波器。DC 系数先进行差分编码再进行熵编码，说明 DC 系数的_____频率分量更多。

解题思路：直接根据差分编码系统的差分方程进行绘图。

完整代码保存在 solved5 中：

```
close all,clear all,clc;
freqz([-1,1],1);
```

作图如下：



根据图像可以看出，该系统是高通滤波器，DC 系数先进行差分编码再进行熵编码，说明 DC 系数的低频分量更多，先通过差分系统滤除低频部分，有利于编码时获得更好的压缩比。

6. DC 预测误差的取值和 Category 值有何关系？如何利用预测误差计算出其 Category？

表 2.2: 亮度直流分量预测误差的 Category 及其 Huffman 编码

预测误差	Category	Huffman 编码
0	0	00
-1,1	1	010
-3, -2, 2, 3	2	011
-7, ···, -4, 4, ···, 7	3	100
-15, ···, -8, 8, ···, 15	4	101
-31, ···, -16, 16, ···, 31	5	110
-63, ···, -32, 32, ···, 63	6	1110
-127 ,···, -64, 64, ···, 127	7	11110
-255, ···, -128, 128, ···, 255	8	111110
-511, ···, -256, 256, ···, 511	9	1111110
-1023, ···, -512, 512, ···, 1023	10	11111110
-2047, ···, -1024, 1024, ···, 2047	11	111111110

根据表格可知，当预测误差为 0 时，Category 的值也为 0，而当预测误差不为 0 时，取预测误差中的最大值，加 1，然后以 2 为底取对数即为 Category 的值，综上，二者可以合并为一个公式 Category=ceil{log2[abs(error)+1]}.

7. 你知道哪些实现 Zig_Zag 扫描的方法？请利用 MATLAB 的强大功能设计一种最佳方法。

解题思路：根据数据与算法课上关于算法高效性的探讨，在数据规模不大的情况下，可以根据 zigzag 的顺序建立相应的表即可。然后通过这个表对需要排序的矩阵进行重新排序即可。（本题的实现一定程度上参考了网上的 zigzag 实现）

代码保存在 zigzag.m 和 solved7.m 文件中，代码如下：

```
function zig_zag = zigzag( input )%实现8*8矩阵的zigzag扫描
zigzag=[1, 2, 9, 17,10,3, 4,11, ...
        18,25,33,26,19,12,5,6 , ...
        13,20,27,34,41,49,42,35,...
        28,21,14,7,8,15,22,29,...
        36,43,50,57,58,51,44,37,...
        30,23,16,24,31,38,45,52,...
        59,60,53,46,39,32,40,47,...
        54,61,62,55,48,56,63,64];
rot=input';
a=reshape(rot,1,64);
temp=a(zigzag);
zig_zag=temp';
end
```

运行结果：

```
in =

   -1.2371   -1.6387   -0.6667    1.1706    0.9199    0.2445    0.4255    0.6003
   -2.1935   -0.7601    0.8641    0.4759    0.1498    0.8084   -1.3147   -1.3615
   -0.3334   -0.8188    0.1134    1.4122    1.4049    0.2130   -0.4164    0.3476
    0.7135    0.5197    0.3984    0.0226    1.0341    0.8797    1.2247   -0.1818
    0.3174   -0.0142    0.8840   -0.0479    0.2916    2.0389   -0.0436   -0.9395
    0.4136   -1.1555    0.1803    1.7013   -0.7777    0.9239    0.5824   -0.0375
   -0.5771   -0.0095    0.5509   -0.5097    0.5667    0.2669   -1.0065   -1.8963
    0.1440   -0.6898    0.6830   -0.0029   -1.3826    0.6417    0.0645   -2.1280

>> result

result =

   -1.2371
   -1.6387
   -2.1935
   -0.3334
   -0.7601
   -0.6667
    1.1706
    0.8641
   -0.8188
```

8.对测试图像分块、DCT 和量化，将量化后的系数写成矩阵的形式，其中每一列为一个块的 DCT 系数 Zig-zag 扫描后形成的列矢量，第一行为各个块的 DC 系数。

解题思路：按照题目要求，先将测试图形进行预处理，然后分块（每块为 8*8 的矩阵），调用 DCT 函数，然后用上一问的 zigzag 函数进行处理即可，由于上一问实现的 zigzag 函数得到的就是 64*1 的列向量，所以直接按列拼接为矩阵即为需要的结果。

完整代码存储在 solved8.m 中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
[leng,wide]=size(hall_gray);
temp=double(hall_gray)-128;  %预处理
Len=leng/8;
Wid=wide/8;
img=zeros(64,Len*Wid);
col=1;
%分块，DCT，量化
for i=1:Len
    for j=1:Wid                %按8*8的规模分块处理
        tempdct=dct2(temp(8*i-7:8*i,8*j-7:8*j));
        tempdct=round(tempdct ./ QTAB);
        img(:,col)=zigzag(tempdct);
        col=col+1;
    end
end
```

运行结果（部分截图）如下：

```
>> img

img =

  1 至 25 列

   57    51     9   -28   -32   -32   -32   -32   -32   -32   -32   -25    27    40    45    50    53
    1     3    22     5     0     0     0     0     0     0     0   -12    -9    -4    -2    -2    -2
    1    -2   -40    -6     0     0     0     0     0     0     0    -5   -26   -12    -5    -1     1
    1     0     0     3     0     0     0     0     0     0     0    -3   -12    -3     0     0    -1
    0     4    -2    -6     0     0     0     0     0     0     0     5    -5    -2    -3    -1     0
   -1    -1     0     3     0     0     0     0     0     0     0    11     0    -1     0     0     0
    0     2     1     1     0     0     0     0     0     0     0    -5     1     0     0     0     0
    0    -2     5    -3     0     0     0     0     0     0     0    -4     2    -1     0     0     0
   -1     2   -12     4     0     0     0     0     0     0     0     4    -3    -1     0     0     0
    0    -2     0    -2     0     0     0     0     0     0     0     1    -5    -1    -1    -1     0
    0    -1     0     1     0     0     0     0     0     0     0     0    -1     0     0     0     0
    0     1     0    -2     0     0     0     0     0     0     0    -1     0     0     0     0     0
    0    -1     0     2     0     0     0     0     0     0     0    -3     1     0     0     0     0
    0     1     0    -1     0     0     0     0     0     0     0     2     1     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     2     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0    -1     0     0     0     0     0
    0    -1     1    -1     0     0     0     0     0     0     0    -1     0     0     0     0     0
    0     1     1     1     0     0     0     0     0     0     0     2    -1     0     0     0     0
    0    -1    -3    -1     0     0     0     0     0     0     0     1     0     0     0     0     0
    0     1     0     1     0     0     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     0     0     0     0     0     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     0     0     0     0     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
    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
```

9.请实现本章介绍的 JPEG 编码（不包括写 JFIF 文件），输出为 DC 系数的码流，AC 系数的码流，图像高度和图像宽度，将这四个变量写入 jpegcodes.mat 文件。

解题思路：1.DC 编码：用到了实验材料中的差分方程，通过表得到的规律可以由预计误差计算出 Category,编码时需要注意对负数取 1 补码（需要注意的是取补码前要先把转化的 2 进制先转化为数字型）。2.AC 编码：根据实验材料中的提示，需要找出相邻两非零数之间的个数，这一问题的解决用到了 MATLAB 中的 find 函数，直接找出符合条件的值的位置，之后容易得到行程，其余部分与 DC 编码的处理类似。

完整的代码保存在 solved9.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('img.mat');
load('JpegCoeff.mat');
[L,W]=size(hall_gray);
image=size(img);
tempDC=img(1,:);
%DC编码
for i=2:image(2)                          %差分编码
    tempDC(i)=img(1,i-1)-img(1,i);
end
DC1=[];
category = ceil(log2(abs(tempDC)+1));  %确定category的值
 for k=1:image(2)
    huffman = DCTAB(category(k)+1, 2:1+DCTAB(category(k)+1,1));
    Magnitude = str2num((dec2bin(abs(tempDC(k))))')';%字符型转换为常数型
    if(tempDC(k)<0) %对负数进行补码处理
       Magnitude=~Magnitude;
    end
     DC1=[DC1,huffman,Magnitude];
 end
%AC编码
AC1=[];
tempAC=img(2:64,:);
EOB=[1,0,1,0];%对块结束编码
ZRL=[1,1,1,1,1,1,1,1,1,0,0,1];
Run=0;%行程
for i = 1:image(2)
    position = find(tempAC(:,i)~=0);%找到非零数的位置
    len=length(position);
    if(len)
        prepos = 1;
        for j = 1:len
            Run = position(j)-prepos;%计算行程
            prepos = position(j) + 1;%更新前一个非0数的坐标
            count=floor(Run/16);      %计算有多少个16个0连续的情况
            Run=Run-16*count;
            while(count>0)
                AC1 = [AC1,ZRL];
```

```
                count = count - 1;
            end
            ac = tempAC(position(j),i);
            Size= ceil(log2(abs(ac)+1));
            Ampli = str2num((dec2bin(abs(ac)))')';
            if(ac<0)
                Ampli = ~Ampli;
            end
            Huffman = ACTAB(Run*10+Size,4:ACTAB(Run*10+Size,3)+3);
            AC1 = [AC1,Huffman,Ampli];
        end
    end
    AC1=[AC1,EOB];%块结束
end
save('jpegcodes.mat','DC1','AC1','L','W');
```

10.计算压缩比（输入文件长度/输出码流长入），注意转换为相同进制。

```
close all,clear all,clc;
load('hall.mat');
load('jpegcodes.mat');
[len,wid]=size(hall_gray);
input=len*wid*8;
output=length(DC1)+length(AC1);
ratio=input/output;
```

计算结果：

```
>> input

input =

      161280

>> output

output =

       25126

>> ratio

ratio =

    6.4188
```

11. 请实现本章介绍的 JPEG 解码，输入是你生成的 jpegcodes.mat 文件。分别用客观（PSNR）和主观方式评价编解码效果如何。

解题思路：根据实验材料的提示，DC 解码先由 Huffman 编码得到 category，然后根据 category 取出 Magnitude，判断符号，转化为 10 进制即可，解码 Huffman 码是利用了给的 DCTAB，通过扫描 DC 与 DCTAB 表中每一行的 huffman 对比，即可取出 category。类似的，AC 解码也可利用 ACTAB 表得到 Run/Size。需要注意的是，ACTAB 中 huffman 码的最长尺度为 16，但是随着扫描的进行，接近尾端时，可能剩余的位数不足 16，导致运行时报错（矩阵维度报错），假设最短是结尾的四位标记 EOB，保险起见，在扫描时在扫描函数中扫描前临时加 12 个 0，保证长度始终大于 16。接下来的步骤和 DC 编码类似，注意根据行程 Run 补上对应个数的 0 即可。解码后通过反 zigzazg 扫描恢复原矩阵即可，需要注意的是需要加上 128。

完整代码保存在 solved11.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');load('img.mat');
load('jpegcodes.mat');
load('JpegCoeff.mat');
Len=L/8;Wid=W/8;
%DCdecode
DC_decode=[];tempDC=DC1;
while(length(tempDC)~=0)
    [huff_len,category]=getcategory(tempDC,DCTAB); %获取category的值
    tempDC(1:huff_len)=[];  %取得category后将对应的huffman编码在tempDC中去除
    if(category==0)
        DC_decode=[DC_decode,0];%直接补0
        tempDC(1)=[];
    else
        Magnitude=tempDC(1:category);%根据category取出对应的Magnitude
        tempDC(1:category)=[];
        if(Magnitude(1)==0) %对负数取补码,并标记符号为负
            Magnitude=~Magnitude;
            signal=-1;
        else
            signal=1;
        end
        error = bin2dec(num2str(Magnitude)); %将Magnitude转化为十进制的预测误差
        if(signal==-1)
            error=-error;
        end
        DC_decode=[DC_decode,error];
    end
end
for i=2:length(DC_decode)
    DC_decode(i)=DC_decode(i-1)-DC_decode(i);    %由差分方程得到解码结果
end
```

```
%ACdecode
EOB=[1,0,1,0];ZRL=[1,1,1,1,1,1,1,1,0,0,1];
tempAC=AC1;col_vector=[];%列向量
AC_decode=[];
ZRLzero=zeros(1,16);
while(length(tempAC)~=0)
    if(tempAC(1:4)==EOB) %当解码到块结束（EOB）时
        addzero=zeros(1,63-length(col_vector)); %直接末尾补0
        Col_vector=[col_vector,addzero]';
        tempAC(1:4)=[]; %已经解码过的都清除
        AC_decode=[AC_decode,Col_vector];
        col_vector=[];%解码下一列前注意清零
    elseif(tempAC(1:11)==ZRL) %当解码到（ZRL）时
        col_vector=[col_vector,ZRLzero]; %补16个0
        tempAC(1:11)=[]; %已经解码过的都清除
    else
        [Run,Size,Huff_len ] = getsizerun(tempAC,ACTAB);
        Runzero=zeros(1,Run);%将行程对应个数的0写入列向量
        col_vector=[col_vector,Runzero];
        tempAC(1:Huff_len)=[];
        Amplitude=tempAC(1:Size);%取出Amplitude
        if(Amplitude(1)==0)    %判断第一位是否为0，对负数取补码
            Amplitude=~Amplitude;
            signal=-1;
        else
            signal=1;
        end
        error = bin2dec(num2str(Amplitude)); %将Amplitude转化为十进制的预测误差
        if(signal==-1)
            error=-error;
        end
        col_vector=[col_vector,error];
        tempAC(1:Size)=[];
    end
end
recovery=zeros(L,W);
re_img=[DC_decode;AC_decode];
Col=1;
for i=1:Len  %反量化
    for j=1:Wid
        rezigzag=re_img(:,Col);
        b=antizigzag(rezigzag);
        Dct=b.*QTAB;
        recovery(8*i-7:8*i,8*j-7:8*j)=idct2(Dct);
        Col=Col+1;
    end
end
recovery=recovery+128;
hall_decode=uint8(recovery);
subplot(1,2,1);imshow(hall_gray);title('原图');
subplot(1,2,2);imshow(hall_decode);title('反编码恢复图');
imwrite(hall_decode,'hall_deode.jpg');
MSE=sum(sum((double(hall_decode)-double(hall_gray)).^2))/L/W;
PSNR=10*log10(255^2/MSE)
```

运行结果：

命令行窗口

```
PSNR =

   31.1874
```

可以看出，与原图相比，反编码恢复的图像更为模糊。进一步，去 hall_gray 反编码得到的 hall_decode 的部分值比较如下图：

命令行窗口

```
>> hall_gray(1:20)

ans =

  248  248  245  243  243  244  239  233  233  234  232  233  235  235  235  235  235  234  234  230

>> hall_decode(1:20)

ans =

  244  244  244  243  243  241  240  240  236  235  235  234  234  233  233  232  236  235  233  229
```

可以看出确实对应的值存在一定的差别。计算得到 PSNR 值为 31.1874，有着较高的相似度。

12.将量化步长减小为原来的一半，重新编解码。同标准量化步长的情况比较压缩比和图像质量。

解题思路，直接将原来的量化步长矩阵改为 QTAB/2 即可。

完整代码保存在 solved12.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
QTAB=QTAB/2;
[leng,wide]=size(hall_gray);
temp=double(hall_gray)-128;  %预处理
Len=leng/8;
Wid=wide/8;
img=zeros(64,Len*Wid);
col=1;
%分块，DCT，量化
for i=1:Len
    for j=1:Wid                %按8*8的规模分块处理
        tempdct=dct2(temp(8*i-7:8*i,8*j-7:8*j));
        tempdct=round(tempdct ./ QTAB);
        img(:,col)=zigzag(tempdct);
        col=col+1;
    end
end
[L,W]=size(hall_gray);
image=size(img);
tempDC=img(1,:);
%DC编码
for i=2:image(2)                          %差分编码
    tempDC(i)=img(1,i-1)-img(1,i);
end
DC1=[];
category = ceil(log2(abs(tempDC)+1));  %确定category的值
 for k=1:image(2)
    huffman = DCTAB(category(k)+1, 2:1+DCTAB(category(k)+1,1));
    Magnitude = str2num((dec2bin(abs(tempDC(k))))')';%字符型转换为常数型
    if(tempDC(k)<0) %对负数进行补码处理
       Magnitude=~Magnitude;
    end
     DC1=[DC1,huffman,Magnitude];
 end
```

```
37      %AC编码
38 -    AC1=[];
39 -    tempAC=img(2:64,:);
40 -    EOB=[1,0,1,0];%对块结束编码
41 -    ZRL=[1,1,1,1,1,1,1,1,1,0,0,1];
42 -    Run=0;%行程
43 -    for i = 1:image(2)
44 -        position = find(tempAC(:,i)~=0);%找到非零数的位置
45 -        len=length(position);
46 -        if(len)
47 -            prepos = 1;
48 -            for j = 1:len
49 -                Run = position(j)-prepos;%计算行程
50 -                prepos = position(j) + 1;%更新前一个非0数的坐标
51 -                count=floor(Run/16);      %计算有多少个16个0连续的情况
52 -                Run=Run-16*count;
53 -                while(count>0)
54 -                    AC1 = [AC1,ZRL];
55 -                    count = count - 1;
56 -                end
57 -                ac = tempAC(position(j),i);
58 -                Size= ceil(log2(abs(ac)+1));
59 -                Ampli = str2num((dec2bin(abs(ac)))')';
60 -                if(ac<0)
61 -                    Ampli = ~Ampli;
62 -                end
63 -                Huffman = ACTAB(Run*10+Size,4:ACTAB(Run*10+Size,3)+3);
64 -                AC1 = [AC1,Huffman,Ampli];
65 -            end
66 -        end
67 -        AC1=[AC1,EOB];%块结束
68 -    end
69 -    Len=L/8;Wid=W/8;
70      %DCdecode
71 -    DC_decode=[];tempDC=DC1;
```

```
while(length(tempDC)~=0)
    [huff_len,category]=getcategory(tempDC,DCTAB); %获取category的值
    tempDC(1:huff_len)=[];  %取得category后将对应的huffman编码在tempDC中去除
    if(category==0)
        DC_decode=[DC_decode,0];%直接补0
        tempDC(1)=[];
    else
        Magnitude=tempDC(1:category);%根据category取出对应的Magnitude
        tempDC(1:category)=[];
        if(Magnitude(1)==0) %对负数取补码,并标记符号为负
            Magnitude=~Magnitude;
            signal=-1;
        else
            signal=1;
        end
        error = bin2dec(num2str(Magnitude)); %将Magnitude转化为十进制的预测误差
        if(signal==-1)
            error=-error;
        end
        DC_decode=[DC_decode,error];
    end
end
for i=2:length(DC_decode)
    DC_decode(i)=DC_decode(i-1)-DC_decode(i);    %由差分方程得到解码结果
end
%ACdecode
EOB=[1,0,1,0];ZRL=[1,1,1,1,1,1,1,1,0,0,1];
tempAC=AC1;col_vector=[];%列向量
AC_decode=[];
ZRLzero=zeros(1,16);
while(length(tempAC)~=0)
    if(tempAC(1:4)==EOB) %当解码到块结束（EOB）时
        addzero=zeros(1,63-length(col_vector)); %直接末尾补0
        Col_vector=[col_vector,addzero]';
        tempAC(1:4)=[]; %已经解码过的都清除
        AC_decode=[AC_decode,Col_vector];
        col_vector=[];%解码下一列前注意清零
    elseif(tempAC(1:11)==ZRL) %当解码到（ZRL）时
```

```
            col_vector=[col_vector,ZRLzero]; %补16个0
            tempAC(1:11)=[]; %已经解码过的都清除
        else
            [Run,Size,Huff_len ] = getsizerun(tempAC,ACTAB);
            Runzero=zeros(1,Run);%将行程对应个数的0写入列向里
            col_vector=[col_vector,Runzero];
            tempAC(1:Huff_len)=[];
            Amplitude=tempAC(1:Size);%取出Amplitude
            if(Amplitude(1)==0)    %判断第一位是否为0，对负数取补码
                Amplitude=~Amplitude;
                signal=-1;
            else
                signal=1;
            end
            error = bin2dec(num2str(Amplitude)); %将Amplitude转化为十进制的预测误差
            if(signal==-1)
                error=-error;
            end
            col_vector=[col_vector,error];
            tempAC(1:Size)=[];
        end
    end
    recovery=zeros(L,W);
    re_img=[DC_decode;AC_decode];
    Col=1;
for i=1:Len  %反量化
    for j=1:Wid
        rezigzag=re_img(:,Col);
        b=antizigzag(rezigzag);
        Dct=b.*QTAB;
        recovery(8*i-7:8*i,8*j-7:8*j)=idct2(Dct);
        Col=Col+1;
    end
end
```

```
recovery=recovery+128;
hall_decode=uint8(recovery);
subplot(1,2,1);imshow(hall_gray);title('原图');
subplot(1,2,2);imshow(hall_decode);title('反编码恢复图');
imwrite(hall_decode,'hall_deode.jpg');
MSE=sum(sum((double(hall_decode)-double(hall_gray)).^2))/L/W;
PSNR=10*log10(255^2/MSE)
```

运行结果如下：

左图为 11 问的解码图，有图为 12 问的解码图。

命令行窗口

```
PSNR =

   34.2067
```

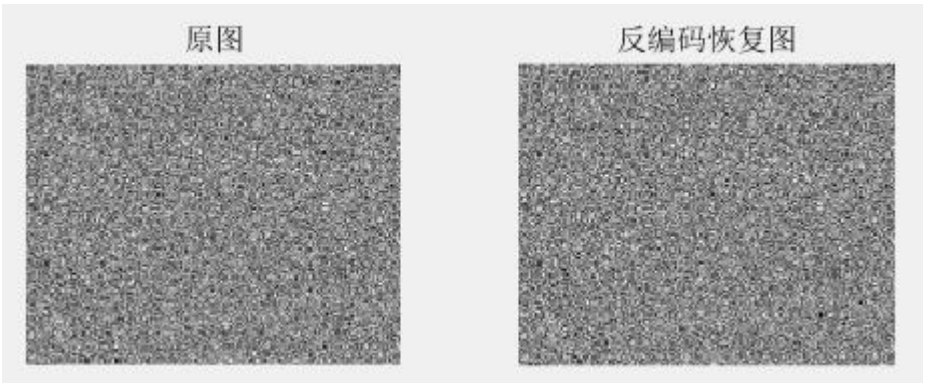
可以看出，直观上，减小量化步长后，解码得到的图像比 11 问的更清晰了。客观上，调用第 10 问的代码得到压缩比约为 4.4，比原来压缩比小，PSNR=34.2067，大于原来的 PSNR 值。进一步验证，牺牲压缩比确实能够使图像失真减少。

13.看电视时偶尔能看到美丽的雪花图像（见 snow.mat），请对其编解码。和测试图像的压缩比和图像质量进行比较，并解释比较结果。

解题思路：代码和 12 问基本一致。

完整代码保存在 solved13.m 文件中，在此不再赘述。

运行结果如下：



命令行窗口

```
PSNR =

   28.3683
```

经过编码和解码后，算得 PSNR=28.3683，明显低于图像的 PSNR 值，进一步调用 10 问的代码算得压缩比为 3.6403，约为 3.6，同样也低于图像的压缩比。观察图像感觉恢复的雪花比前面的更模糊了。根据编码原理，由于雪花的高频分量明显多于普通图像，更多的高频分布导致量化后非 0 部分增多，压缩比降低，此外由于编码原理决定高平分量损失较多，误差较大，高频分量较多的图片使用 JPEG 编码自然会有较大失真。

三 、信息隐藏

练习题：

1.实现本章介绍的空域隐藏方法和提取方法。验证其抗 JPEG 编码能力。

解题思路：空域隐藏方法和提取方法的实现较为简单，首先将需要隐藏的信息转化为二进制码流，用每一位替换掉原图像像素点的二进制数的最低位即可。提取是一个逆向过程，先对原图像进行扫描，取出隐藏信息的像素点的二进制的最低位，然后将得到的二进制转换为字符串，即可得到原来的信息。本题中隐藏的信息是（MATLAB is very interesting!）.JPEG 编解码直接利用上一题的 12 问即可。

完整代码保存在 solved1.m 文件中，代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
subplot(1,2,1);imshow(hall_gray);title('原图');
doc='MATLAB is very interesting!'; %输入文字
data=dec2bin(double(doc));%将需要隐藏的信息转化为二进制码流
ascii_code=reshape(data',1,numel(data));%由numel函数得到矩阵的元素个数
ascii_len=length(ascii_code);
count=1;
for i=1:size(hall_gray,1)
    for j=1:size(hall_gray,2)
        if(count<=ascii_len)
            hall_bin=dec2bin(hall_gray(i,j));%替换低位为隐藏信息
            hall_gray(i,j)=hall_gray(i,j)-hall_bin(end)+ascii_code(count);
            count=count+1;
        end
    end
end
%run JPEG;%JPEG编解码
%获取隐藏信息
count=1;
getinfo=zeros(1,ascii_len);
for i=1:size(hall_gray,1)
    for j=1:size(hall_gray,2)
        if(count<=ascii_len)
            hall_bin=dec2bin(hall_gray(i,j));%替换低位为隐藏信息
            getinfo(count)= hall_bin(end)-'0';%获取信息
            count=count+1;
        end
    end
end
info=[];
for i=1:length(doc)
     info2str=num2str(getinfo(7*i-6:7*i));
     tempinfo=bin2dec(info2str);
     info=[info,tempinfo];
end
subplot(1,2,2);imshow(hall_gray);title('加入隐藏信息后的图');
char(info)
```

运行结果：
未进行 JPEG 编解码时：

还原的效果：

```
命令行窗口

ans =

MATLAB is very interesting!
```

进行 JPEG 编解码后：

```
命令行窗口

ans =

  %eFX          k    @H  K>v      h"
```

可以看出，图片加入隐藏信息后，仅有一点模糊，并没有大范围的失真，结合原理可知，隐藏的信息替换的是像素点的低位，对图像的影响较小。但是经过 JPEG 编解码后，结果完全不对，说明空域隐藏的抗 JPEG 编码能力很低，使用本方法隐藏信息不能再用 JPEG 进行图片压缩。

2. 依次实现本章介绍的三种变换域信息隐藏方法与提取方法，分析嵌密方法的隐蔽性以及嵌密后 JPEG 图像的质量变化和压缩比变化。

（1）解题思路：该方法与空域隐藏方法比较像，在第一问的基础上改动即可，将隐藏信息放在 DCT 后，编码之前，将获取信息放在解码后，反 DCT 变换之前即可。

完整代码保存在 solved2_1.m 中，部分关键代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
subplot(1,2,1);imshow(hall_gray);title('原图');
doc='good!'; %输入文字
data=dec2bin(double(doc));%将需要隐藏的信息转化为二进制码流
ascii_code=reshape(data',1,numel(data));%由numel函数得到矩阵的元素个数
ascii_len=length(ascii_code);
%JPEG编解码
```

```
%信息隐藏
count=1;
for i=1:size(img,1)
    for j=1:size(img,2)
        if(count<=ascii_len)
            if(img(i,j)<0)
                img_bin=dec2bin(abs(img(i,j)));%替换低位为隐藏信息(负数则取绝对值)
            else
                img_bin=dec2bin(img(i,j));%替换低位为隐藏信息
            end
            img(i,j)=img(i,j)-img_bin(end)+ascii_code(count);
            count=count+1;
        end
    end
end
```

```
%获取隐藏信息
count=1;
getinfo=zeros(1,ascii_len);
for i=1:size(re_img,1)
    for j=1:size(re_img,2)
        if(count<=ascii_len)
            if(re_img(i,j)<0)
                reimg_bin=dec2bin(abs(re_img(i,j)));%替换低位为隐藏信息(负数去绝对值)
            else
                reimg_bin=dec2bin(re_img(i,j));%替换低位为隐藏信息
            end
            getinfo(count)= reimg_bin(end)-'0';%获取信息
            count=count+1;
        end
    end
end
info=[];
for i=1:length(doc)
     info2str=num2str(getinfo(7*i-6:7*i));
     tempinfo=bin2dec(info2str);
     info=[info,tempinfo];
end
char(info)
```

运行结果如下：

原图　　加入隐藏信息后的图

命令行窗口

```
ans =

good!
```

可以看出，利用此方法，即使经过 JPEG 编码后，依然能够得到正确的隐藏信息（good!），加入信息后图片也无较大的失真，由于信息量较小，放大后仅仅有部分像素块灰度变化与周围有一定的不连续，这可能是由于对 DCT 系数进行低位替换后，导致原像素点系数有一定变化，导致了还原的图片对应 像素点的亮度变化。

（2）解题思路：与第一题差不多，只是对一部分进行信息替换，例如只选取 8*8 采样区域 DCT 系数的前几位加入隐藏信息，这里由于文本量较小，只选取 DC（直流）系数来隐藏信息。完整代码保存在 solved2_2.m 文件中，关键代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
subplot(1,2,1);imshow(hall_gray);title('原图');
doc='good!'; %输入文字
data=dec2bin(double(doc));%将需要隐藏的信息转化为二进制码流
ascii_code=reshape(data',1,numel(data));%由numel函数得到矩阵的元素个数
ascii_len=length(ascii_code);
```

```
%信息隐藏
count=1;
for j=1:size(img,2)%只选取左上角的一小部分进行编码
        if(count<=ascii_len)
           if(img(1,j)<0)
               img_bin=dec2bin(abs(img(1,j)));%替换低位为隐藏信息(负数则取绝对值)
            else
               img_bin=dec2bin(img(1,j));%替换低位为隐藏信息
            end
            img(1,j)=img(1,j)-img_bin(end)+ascii_code(count);
            count=count+1;
        end
 end
```

```
%获取隐藏信息
count=1;
getinfo=zeros(1,ascii_len);
for j=1:size(re_img,2)
      if(count<=ascii_len)
         if(re_img(1,j)<0)
            reimg_bin=dec2bin(abs(re_img(1,j)));%替换低位为隐藏信息(负数去绝对
         else
            reimg_bin=dec2bin(re_img(1,j));%替换低位为隐藏信息
         end
         getinfo(count)= reimg_bin(end)-'0';%获取信息
         count=count+1;
      end
end
info=[];
for i=1:length(doc)
      info2str=num2str(getinfo(7*i-6:7*i));
      tempinfo=bin2dec(info2str);
      info=[info,tempinfo];
end
char(info)
```

运行结果：

可以看出，该方法抗 JPEG 编码能力依旧不错，能得到正确的答案。由于是对 DC 系数（非 0）加入信息，像素点变化也不明显，放大看也基本没什么失真，与原图相比也是 JPEG 引起的正常压缩导致的图像模糊。但是选取部分非 0 的 DCT 系数加入信息，会导致图片的信息可隐藏量大大降低。

（3）：追加在 zigzag 序列的最后一个非零为后面。

解题思路：直接用 MATLAB 自带的 find 函数即可找出每一列 zigzag 扫描向量的最后一个非 0 数，按照实验材料的说明，判断对应的位置是否为 64，如果是，则直接替换为信息位，否则将下一位替换为信息位，获取信息更简单，直接用 find 函数找出最后一位非 0 位即为所需信息。

完整代码保存在 solved2_3.m 文件中，关键代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
subplot(1,2,1);imshow(hall_gray);title('原图');
Doc='good!'; %输入文字
data=dec2bin(double(Doc))-'0';%将需要隐藏的信息转化为二进制码流
ascii_code=reshape(data',1,numel(data));%由numel函数得到矩阵的元素个数
ascii_code(ascii_code==0)=-1;%将二进制中的0换为-1
ascii_len=length(ascii_code);
```

```
%信息隐藏
count=1;
for j=1:size(img,2)%只选取左上角的一小部分进行编码
        if(count<=ascii_len)
            img_pos=find(img(:,j)~=0,1,'last');%获取每一列最后一个非0数的位置
            if(img_pos==64)
                img(img_pos,j)=ascii_code(count);%替换为隐藏信息
             else
                img(img_pos+1,j)=ascii_code(count);%非零的下一位替换为信息位
             end
             count=count+1;
        end
 end
```

```
153      %获取隐藏信息
154 -    count=1;
155 -    getinfo=zeros(1,ascii_len);
156 -    for j=1:size(re_img,2)
157 -        if(count<=ascii_len)
158 -            reimg_pos=find(re_img(:,j)~=0,1,'last');%获取每一列最后一个非0数的位置
159 -            if(re_img(reimg_pos,j)==-1)
160 -                getinfo(count)=0;
161 -            else
162 -                getinfo(count)= re_img(reimg_pos,j);%获取信息
163 -            end
164 -            count=count+1;
165 -        end
166 -    end
167 -    info=[];
168 -    for i=1:length(Doc)
169 -        info2str=num2str(getinfo(7*i-6:7*i));
170 -        tempinfo=bin2dec(info2str);
171 -        info=[info,tempinfo];
172 -    end
173 -    char(info)
174      %反里化
```

运行结果：

可以看出，图片基本没什么失真，且获取的信息也正确。相比于前两种方法，本方法更为简单，无须对 DCT 系数进行二进制转化，替换低位，而是选取最后一个非 0 位的下一位直接替代即可，在操作上更为简单，但是同样也是以牺牲大量信息隐藏量为前提的。

综上所述：方法逐一嵌入信息，密度过高会导致细微的失真，方法二、三注意保持一定间隔进行信息隐藏，图像保持更好。但就可隐藏信息量而言，方法二、三远不如方法一。所以在实际运营过程中，二者需要去一个比较好的平衡，既要尽可能多地隐藏信息，又要避免图片失真。

3.（选做）请设计实现新的隐藏算法并分析其优缺点。

算法设计：根据前面的方法可知，对 DCT 系数信息隐藏时替换低位导致的 DCT 系数变动应该为 1，但是 DCT 系数如果本身比较小，则影响就较大。所以我们可以根据需要隐藏的信息量大小，人为规定一个阈值，只对整个图像 DCT 系数中大于阈值的部分进行信息隐藏操作，这样也能一定程度上减少失真。

解题思路：根据自己设计的算法，直接对 DCT 系数矩阵进行全盘扫描，由于信息量较小，所以认为规定 DCT 系数绝对值阈值为 40，只对 DCT 系数矩阵中绝对值大于等于 40 的进行信息隐藏操作。即保证了信息隐藏的稀疏，又保证了替换后对原 DCT 系数的影响尽可能小。

完整代码保存在 solved3.m 文件中，关键代码如下：

```
close all,clear all,clc;
load('hall.mat');
load('JpegCoeff.mat');
subplot(1,2,1);imshow(hall_gray);title('原图');
doc='good!'; %输入文字
data=dec2bin(double(doc));%将需要隐藏的信息转化为二进制码流
ascii_code=reshape(data',1,numel(data));%由numel函数得到矩阵的元素个数
ascii_len=length(ascii_code);
```

```
%信息隐藏
count=1;
for i=1:size(img,1)
    for j=1:size(img,2)
        if(abs(img(i,j)>=40))%设置阈值
            if(count<=ascii_len)
              if(img(i,j)<0)
                 img_bin=dec2bin(abs(img(i,j)));%替换低位为隐藏信息(负数则取绝对值)
              else
                 img_bin=dec2bin(img(i,j));%替换低位为隐藏信息
              end
              img(i,j)=img(i,j)-img_bin(end)+ascii_code(count);
              count=count+1;
            end
        end
    end
end
```

```
%获取隐藏信息
count=1;
getinfo=zeros(1,ascii_len);
for i=1:size(re_img,1)
    for j=1:size(re_img,2)
        if(abs(re_img(i,j)>=40))
           if(count<=ascii_len)
             if(re_img(i,j)<0)
                reimg_bin=dec2bin(abs(re_img(i,j)));%替换低位为隐藏信息(负数去绝对值)
             else
               reimg_bin=dec2bin(re_img(i,j));%替换低位为隐藏信息
             end
             getinfo(count)= reimg_bin(end)-'0';%获取信息
             count=count+1;
           end
        end
    end
end
info=[];
for i=1:length(doc)
     info2str=num2str(getinfo(7*i-6:7*i));
     tempinfo=bin2dec(info2str);
     info=[info,tempinfo];
end
char(info)
```

运行结果：

从结果看出，此方法确实能够得到正确的隐藏信息，由于是根据输入文本（good!）的二进制码长度设置的阈值，保证了替换信息对被替换 DCT 系数的影响最小，同时也仿真信息隐藏过于密集，所以图片无明显失真。但缺点也比较明显，需要根据输入文本量的大小人为调整阈值， 阈值设置过大，可能导致可隐藏位不足，信息无法完全隐藏，恢复的信息也可能不完整。

四、人脸检测

练习题：

1. 所给资料 Faces 目录下包含从网图中截取的 33 张人脸，试以其作为样本训练人脸标准 v。

（a）样本人脸大小不一，是否需要首先将图片调整为相同大小？

不需要，算法中要求的是获得各种颜色出现的频率，与图片的尺寸大小无关。

（b）假设 L 分别取 3，4，5，所得三个 v 之间有何关系？

解题思路：根据实验材料，先对提供的图片进行分区域（由 L 决定），得到每个区域各种颜色出现的频率作为特征，然后由特征求出平均值 v 即可。

完整代码保存在 facetest1.m 文件中，代码如下：

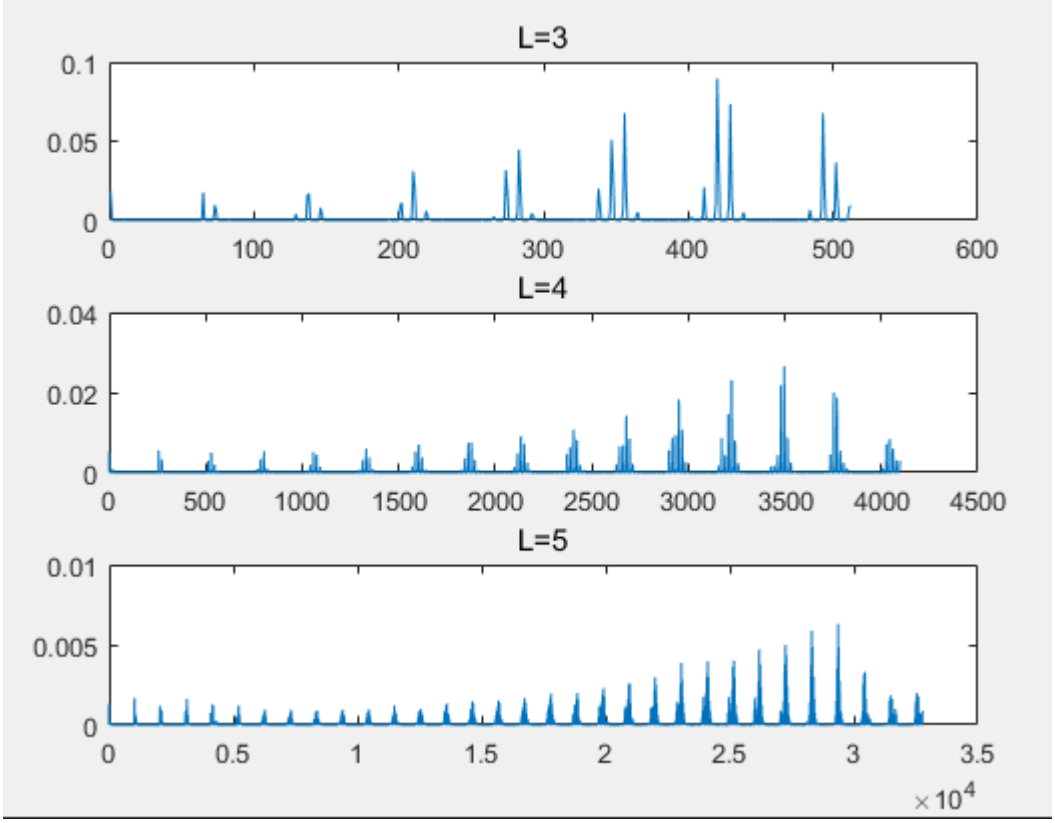
根据材料中的判定方法，得到各颜色出现的频率作为特征，实现该功能的函数如下：

```
function feature = getfeature(img,L)
    [len,wide,high] = size(img);
    feature = zeros(1,2^(3*L));%初始化
    range = 2^(8 - L); %计算每种颜色覆盖的数据范围
    image=floor(double(img)/range);
    NUM=image(:,:,1)*2^(2*L)+image(:,:,2)*2^L+image(:,:,3)+1;
    for i=1:len
        for j=1:wide
            feature(NUM(i,j))=feature(NUM(i,j))+1; % 得到图像中各颜色出现的次数
        end
    end
    feature = feature/(len*wide); % 计算各颜色出现的频率
end
```

对图片特征取平均值，得到不同 L 值下的 v,将结果保存在 mat 文件中。

```
close all,clear all,clc;
L=[3,4,5];imgnum=33;
v1=zeros(1,2^(3*L(1)));
v2=zeros(1,2^(3*L(2)));
v3=zeros(1,2^(3*L(3)));
for count=1:imgnum
    img=imread([num2str(count),'.bmp']);
    v1=v1+getfeature(img,L(1))/imgnum;
    v2=v2+getfeature(img,L(2))/imgnum;
    v3=v3+getfeature(img,L(3))/imgnum;
end
subplot(3,1,1);plot(v1);title('L=3');
subplot(3,1,2);plot(v2);title('L=4');
subplot(3,1,3);plot(v3);title('L=5');
save('feature.mat','v1','v2','v3');
```

运行结果：



通过作图，可以发现随着 L 的增大，峰的数目增加，且包络趋向于正态分布，这是由于随着 L 的增加，颜色数目增加，量化更加细化。

2. 设计一种从任意大小的图片中检测任意多张人脸的算法并编程实现（输出图像在判定为人脸的位置加上红色的方框）。随意选取一张多人照片（比如支部活动或者足球比赛），对程序进行测试。尝试 L 分别取不同的值，评价测试结果有何区别。

解题思路：按照实验材料的提示，首先将待识别图片分区域利用上一问的方法计算 v 值，与训练得到的数据进行比较，如果小于事先设定的阈值，则将此区域标记位可能是人脸的区域，由标记再确定区域的具体坐标的，然后就是对这些区域重叠部分进行合并，具体方法是对两个比较的区域对应点的坐标相减，如果差值大于前面一个的长（或者宽），则没有重叠，否则有重叠。对于重叠的处理，比较方便的是用重叠最外围的坐标替换原坐标。最后一步就是在原图上将这些区域的边用红线画出即可。

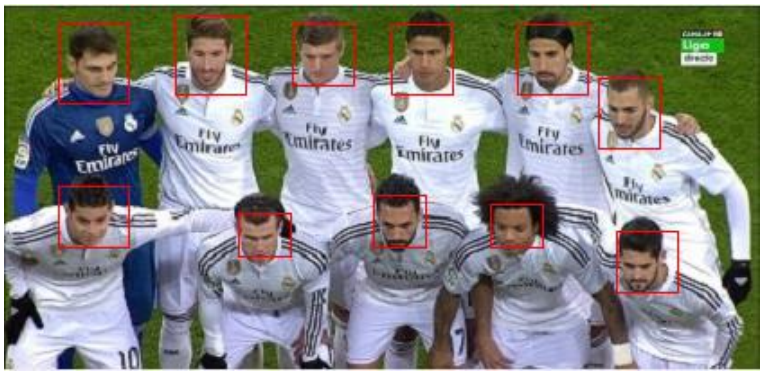
完整代码保存在 facetest2.m 文件中，主要代码如下：

```
close all,clear all,clc;
img=imread('sample1.jpg');
load('feature.mat');
[len,wide,high]=size(img);
L=3;v=v1;e=0.46;%设定阈值
step_len=5;%初始化步长
sepa_len=5*step_len;
mayface=zeros(len,wide);
isface=[];
for i=1:step_len:len-sepa_len
    for j=1:step_len:wide-sepa_len
        separate=img(i:i-1+sepa_len,j:j-1+sepa_len,:);
        u=getfeature(separate,L);
        distance=1-sum(sqrt(v.*u)); %计算与与人脸标准的距离
        if(distance<e)              %小于阈值，则可以判别为人脸
            mayface(i,j)=1;%对可能是人脸的进行标记
        end
    end
end
%通过标记获取坐标
for i=1:len
    for j=1:wide
        if(mayface(i,j)==1)
            isface=[isface,[i,i-1+sepa_len,j,j-1+sepa_len]'];%获取可能是人脸的坐标
        end
    end
end
```

```
m=1;
while(m<length(isface)) %合并重叠部分
    for n=m+1:length(isface)
        L_len=min(abs(isface(2,m)-isface(1,m)),abs(isface(2,n)-isface(1,n)));
        W_len=min(abs(isface(4,m)-isface(3,m)),abs(isface(4,n)-isface(3,n)));
        flag=(abs(isface(1,m)-isface(1,n))>=L_len)||(abs(isface(3,m)-isface(3,n))>=W_len);
        if(~flag)
            isface(1,m)=min(isface(1,m),isface(1,n));
            isface(2,m)=max(isface(2,m),isface(2,n));
            isface(3,m)=min(isface(3,m),isface(3,n));
            isface(4,m)=max(isface(4,m),isface(4,n));
            isface(:,n)=isface(:,m);  %用合并的新坐标替换之前比较的两个区域的坐标
        end
    end
    m=m+1;
end
```

```
44 -    m=1;%将识别的区域用红框框出
45 -    while(m<length(isface))
46 -        for k=isface(1,m):isface(2,m)
47 -            img(k,isface(3,m),:)=[255,0,0];
48 -            img(k,isface(4,m),:)=[255,0,0];
49 -        end
50 -        for l=isface(3,m):isface(4,m)
51 -            img(isface(1,m),l,:)=[255,0,0];
52 -            img(isface(2,m),l,:)=[255,0,0];
53 -        end
54 -       m=m+1;
55 -    end
56 -    imshow(img);
57 -    imwrite(img,'facetest.jpg');
```

L=3 时，阈值取 0.46，得到的效果较好：



L=4 时，阈值取 0.58，能得到比较好的效果：

L=5 时，阈值取 0.74，能得到较好的效果：

对于不同的 L 值需要取不同的阈值才能保证得到较好的识别效果。且 L 和阈值的改变，与图中人脸面部的识别位置（红框位置）还是有些许的变化。

3.对上述图像分别进行如下处理后

（a）顺时针旋转 90°（imrotate）；

（b）保持高度不变，宽度拉伸为原来的 2 倍（imresize）；

（c）适当改变颜色（imadjust）；

再试试你的算法检测结果如何？并分析所得结果。

（a）顺时针旋转 90°（imrotate）：

可以看出，旋转 90 度对人脸识别算法几乎没影响，人能较为准确地识别人脸位置。

（b）保持高度不变，宽度拉伸为原来的 2 倍（imresize）；

可以看出改变宽度还是有一定影响，第一排左二的人脸识别位置不是很准确，有一定的偏移。

（c）适当改变颜色（imadjust）；

可以看出如果过分改变图片颜色，实验指导书提供的人脸识别算法将完全失效。由于本算法完全依赖颜色来识别人脸，训练好的数据中人脸的颜色和颜色改变后的人脸的距离自然不在设定阈值内，所以无法识别，这也是以颜色为基础的人脸检测算法的弊端。

4.如果可以重新选择人脸样本训练标准，你觉得应该如何选取？
根据上一题第三小问的结果，我们知道实验指导书提供的人脸识别算法对于颜色的依赖度很高。所以训练时我们 最大程度地找出人脸有别与其他事物，甚至是有别于人自身其他部位的特征。所以为了得到较为准确的面部特征，首先就是的加大训练的样本量，本实验还是太少，并且需要涵盖不同年龄，不同人种，不同角度，不同面部表情以及不同的光照条件等。由于不同人种之间肤色差异较大，最好分开训练，但如果同一图片中有不同人种（例如白人和黑人均有），识别难度还是较大，这也是基于颜色的人脸检测算法的弊端，此类检测还需借由其他算法辅助才行。

原创性说明：

本次实验绝大部分内容均由本人独立完成，仅 zigzag 扫描方法参考了网上的算法，人脸检测部分由于对于加框前重叠部分的合并，自己的算法不是很有效，参考了版本的做法，但具体实现有一定的差异，而且对于重叠的判断条件，版本的做法是直接取两个区域第二个区域的边长，但如果是大框包含小框的情况可能会出现问题，由于第一个区域是随着扫描在合并，肯定是大于待比较区域，所以我的判别条件是如果对应点的坐标差小于比较的两区域中较大者的边长，就判别为重叠，更为合理。

代码清单说明：

提交的代码文件按照四个大题，分别存在 chapter1~chapter4 文件夹中。

实验感想：

本次实验相比于实验一语音处理更为复杂，通过本次实验，我对图像处理有了一个初步的认识。第一部分讲解了图像处理的一些基本处理方法；第二部分主要讲了 JPEG 编解码，这一部分是实验的重难点，耗时较多，通过本部分的联系，我对 JPEG 编码的具体方法有了比较深刻的理解；第三部分是信息隐藏，其实在第二部分的基础上，第三部分还是比较容易，信息隐藏也比较有意思。最后一部分就是比较有意思的人脸检测环节，人脸检测是个听起来比较高大上的话题，实验材料中提供的是基于颜色的人脸检测方法，原理还是比较简单的，实现起来也不复杂，训练和通过阈值找出待定的人脸区域都不是很复杂，反而正在合并重叠区域的算法设计上花费了较多的时间。由于自己刚开始设计的方法并不能完全合并重叠区域，所以也参考了版本的做法。但是版本关于是否覆盖的判断也不是很合理，替换掉步长和划分区域的边长数据后，发现也不能完全合并重叠。在此基础上，我重新找出了更为合理的重叠判断标准，合并过程也是采用自己的方法实现。

总之，本次实验在巩固 MATLAB 语言知识的同时，也让我对图像处理有了一个初步认识，为后续课程打下基础。