

图像处理大作业

无04 2019012137 张鸿琳

(如果需要运行代码, 请将所需文件根据代码放在相应位置)

第一章练习题

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;
- (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两个图, 看是否达到了目标。

该题目比较容易实现, 只要改变满足特定条件的像素点的RGB值即可。对于(a), 只要判定像素点与图像中心的距离即可, 当小于半径时, 就将其RGB值改为(255,0,0); 对于(b), 则只需要判断像素点处于哪个格子即可, 将其中满足条件的格子中的像素点的RGB值改为(0,0,0)。根据该思路, 编写代码如下 (即 hw_1_2.m 文件) :

```
1 clear all; close all; clc;
2
3 load("hall.mat");%读取文件
4 imwrite(hall_color,"hw_1_2_1.jpg");%输出原图片
5
6 %给测试图像中心加上一红色实心圆
7 test1=hall_color;
8 a=size(hall_color,1);
9 b=size(hall_color,2);
10 R=min(a,b);
11 for i=1:a
12     for j=1:b
13         %若像素点位置位于圆区域内, 则改变其RGB值为纯红色
14         if((i-a/2)^2+(j-b/2)^2<=R^2/4)
15             test1(i,j,1)=255;
16             test1(i,j,2)=0;
17             test1(i,j,3)=0;
18         end
19     end
20 end
21 imwrite(test1,"hw_1_2_2.jpg");%输出画上圆的图像
22
23 %将测试图像涂成"黑白格"的样子
24 test2=hall_color;
25 num=8;%行和列的分格数
26 x=a/num;
27 y=b/num;
28 for i=1:a
29     for j=1:b
30         %格数(行或列)每变动一个, 则黑白发生一次变化
31         if(mod(floor((i-1)/x)+floor((j-1)/y),2)==0)
32             test2(i,j,1)=0;
33             test2(i,j,2)=0;
34             test2(i,j,3)=0;
```

```

35         end
36     end
37 end
38 imwrite(test2,"hw_1_2_3.jpg");%输出化为"黑白格"的图像

```

本来的测试图像如下：



原测试图像(hw_1_2_1.jpg)

在中心画上红色实心圆后的图像如下：



画上红色实心圆后的图像(hw_1_2_2.jpg)

涂成“黑白格”后的图像（分成了8×8格）如下：



涂成“黑白格”后的图像(hw_1_2_3.jpg)

可以看到，该代码成功实现了 (a) 和 (b) 中要求的效果。

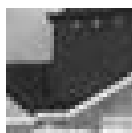
第二章练习题

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。

该步骤可以在变换域进行，因为 DCT 变换就是在空域矩阵上分别左乘和右乘一个矩阵，即 $C = DPD^T$ ，那么在空域减去一个元素值全为 128 的矩阵 X ，就相当于在变换域减去矩阵 DXD^T ，而 DXD^T 只含有一个非零元素，即其第一行第一列的元素，其大小视测试图块的大小而定，其实也就对应于 DCT 变换后矩阵中的直流分量。按照该思路编写代码如下（即 hw_2_1.m 文件）：

```
1 clear all; close all; clc;
2
3 load("hall.mat");%读取图像信息
4
5 %空域实现
6 test1=double(hall_gray(32:63,32:63));
7 imwrite(uint8(test1),"hw_2_1_1.jpg");%输出截取的一块原图
8 test1=test1-128*ones(32,32);
9 imwrite(uint8(test1),"hw_2_1_2.jpg");%输出空域操作后图像
10
11 %变换域实现
12 test2=double(hall_gray(32:63,32:63));
13 C=dct2(test2);%求解变换域矩阵
14 C(1,1)=C(1,1)-4096;%直流分量减去4096，因为测试图块大小为32×32
15 test2=idct2(C);%还原到空域
16 imwrite(uint8(test2),"hw_2_1_3.jpg");%输出变换域操作后图像
17
18 differ=test1-test2 %比较空域实现和变换域实现的差异
```

其中截取的一块原图如下：



截取的一块原图(hw_2_1_1.jpg)

对其进行空域处理后，得到图像如下：



空域处理后的图像(hw_2_2_2.jpg)

对其进行变换域处理后，再恢复出空域信息，得到图像如下：



变换域处理后的图像(hw_2_2_3.jpg)

空域和变换域实现的结果差异在 10^{-13} 量级，可以忽略。

```

differ =

1.0e-13 *

列 1 至 12

0.2132    -0.1421    -0.1421         0         0    -0.1421    -0.1421    0.1421    -0.1421         0    -0.5684    -0.1421
0.3553    0.4263    0.2842    0.1421    0.2842    0.0711    -0.0711    0.0711    -0.0711    0.4263    -0.1421    0.3197
0.4263    0.2132    -0.0711    -0.1421    -0.1421    -0.0711    -0.0711    0.1421         0    0.2842    -0.2842    0.2842
0.2132    0.2842    0.2842         0    -0.1421    -0.2132    -0.0711    0.1421    -0.1421    0.1421    -0.4263    0.0711
0.1421         0         0    0.2132    0.0711    -0.2132    -0.2132    0.0711    -0.0711    -0.1421    -0.4263    0.1421
0.4263    0.1421    0.1421    0.1421    0.1421    0.0711    -0.0711    0.2132    0.0711    0.0711    -0.3553    0.1066
-0.0711    0.0711    -0.0711    -0.4263    -0.4263    -0.0711    -0.2132    -0.1421    -0.4263         0    -0.5684    -0.1421
0.2132    0.1421    0.1421    0.0711    -0.0711    -0.2132    -0.0711    0.0711    -0.2132    0.2132    -0.2132    0.1421
0.2842    0.0711    -0.0711    -0.2132    -0.3553    -0.2132    -0.0711    -0.2132    -0.3553    -0.0711    -0.3553    -0.1421
0.4263         0         0    0.0711    0.0711    0.2132    0.2132    0.1421    -0.1421    0.3553    0.2132    0.2842
-0.0711    -0.1421    -0.1421    -0.1421    -0.2842    -0.1421    -0.1421    -0.2132    -0.3553    -0.1421    -0.4263    -0.0711
0.3553    0.2842    0.2842    0.1421    0.1421    0.4263    0.4263    0.0711    -0.0711    0.2132    0.0711    0.1421

```

空域实现和变换域实现的结果差异

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。

根据实验指导书中的理论推导，二维 DCT 本质上就是对空域矩阵做了两次矩阵乘法，即 $C = DPD^T$ 。按照该思路，编写代码如下（即 hw_2_2.m 文件）：

```

1 clear all; close all; clc;
2
3 load("hall.mat");%读取图像信息
4
5 test=double(hall_gray(1:8,1:8));%取出一块区域数据用于测试
6 test=test-128*ones(8,8);%灰度值减去128
7 C1=myDCT(test);%利用myDCT函数进行DCT变换计算
8 C2=dct2(test);%利用dct2函数进行DCT变换计算
9 differ=C1-C2 %比较两个结果的差异
10
11
12 %进行DCT变换
13 function C=myDCT(P)
14 N=size(P,1);
15 D=zeros(N,N);
16
17 %生成D矩阵
18 for i=1:N
19     for j=1:N
20         if(i==1)
21             D(i,j)=(1/2)^0.5;
22         else
23             D(i,j)=cos((i-1)*(2*j-1)*pi/(2*N));
24         end
25     end
26 end
27 D=(2/N)^0.5*D;
28
29 %得到变换域矩阵
30 C=D*double(P)*D.';
31 end

```

运行代码，得到 myDCT 函数和 dct2 函数的计算相差如下：

differ =

1.0e-12 *

0.2274	-0.0195	-0.0853	0.0786	0.0795	0.2685	-0.2318	-0.1876
0.0213	-0.0187	0.0131	0.0027	0.0027	-0.0052	-0.0268	0.0089
-0.0941	-0.0018	-0.0022	0.0080	0.0040	-0.0024	-0.0018	0.0016
0.0624	-0.0044	-0.0112	-0.0079	-0.0002	-0.0036	0.0024	0.0032
0.0311	-0.0018	-0.0013	0.0013	-0.0098	0.0057	-0.0039	-0.0004
0.2746	-0.0020	-0.0013	0.0056	-0.0091	0.0022	-0.0075	-0.0051
-0.1767	0.0024	-0.0022	-0.0011	0.0068	-0.0098	0.0050	-0.0029
-0.1016	0.0103	-0.0124	0.0167	-0.0224	0.0243	-0.0198	0.0099

myDCT 函数和 dct2 函数结果差异

可以看到，二者计算结果几乎没有差别。MATLAB 自带 dct2 函数源代码如下：

```
1 function b=dct2(arg1,mrows,ncols)
2 %DCT2 2-D discrete cosine transform.
3 % B = DCT2(A) returns the discrete cosine transform of A.
4 % The matrix B is the same size as A and contains the
5 % discrete cosine transform coefficients.
6 %
7 % B = DCT2(A,[M N]) or B = DCT2(A,M,N) pads the matrix A with
8 % zeros to size M-by-N before transforming. If M or N is
9 % smaller than the corresponding dimension of A, DCT2 truncates
10 % A.
11 %
12 % This transform can be inverted using IDCT2.
13 %
14 % Class Support
15 % -----
16 % A can be numeric or logical. The returned matrix B is of
17 % class double.
18 %
19 % References
20 % -----
21 % 1) A. K. Jain, "Fundamentals of Digital Image Processing", pp. 150-153.
22 % 2) wallace, "The JPEG Still Picture Compression Standard",
23 % Communications of the ACM, April 1991.
24 %
25 % Example
26 % -----
27 % RGB = imread('autumn.tif');
28 % I = rgb2gray(RGB);
29 % J = dct2(I);
30 % imshow(log(abs(J)),[]), colormap(gca,jet), colorbar
31 %
32 % % The commands below set values less than magnitude 10 in the
33 % % DCT matrix to zero, then reconstruct the image using the
34 % % inverse DCT function IDCT2.
35 %
36 % J(abs(J)<10) = 0;
37 % K = idct2(J);
38 % figure, imshow(I)
39 % figure, imshow(K,[0 255])
40 %
```

```

41 % See also FFT2, IDCT2, IFFT2.
42
43 % Copyright 1992-2019 The MathWorks, Inc.
44
45 matlab.images.internal.errorIfgpuArray(arg1);
46 [m, n] = size(arg1);
47 % Basic algorithm.
48 if (nargin == 1)
49     if (m > 1) && (n > 1)
50         b = dct(dct(arg1).').';
51         return;
52     else
53         mrows = m;
54         ncols = n;
55     end
56 end
57
58 % Padding for vector input.
59 a = arg1;
60 if nargin==2, ncols = mrows(2); mrows = mrows(1); end
61 mpad = mrows; npad = ncols;
62 matlab.images.internal.errorIfgpuArray(mrows,ncols);
63
64 if m == 1 && mpad > m, a(2, 1) = 0; m = 2; end
65 if n == 1 && npad > n, a(1, 2) = 0; n = 2; end
66 if m == 1, mpad = npad; npad = 1; end % For row vector.
67
68 % Transform.
69
70 b = dct(a, mpad);
71 if m > 1 && n > 1, b = dct(b.', npad).'; end

```

MATLAB 自带的 `dct2` 函数包含了更多对输入数据正确性的检查报错机制，其求解思路依托于 `dct` 函数，也就是一维 DCT 变换，这和我直接进行二维矩阵运算得到 DCT 结果的思路还是有所差异的。

3. 如果将 DCT 系数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。如果左侧的四列置零呢？

DCT 系数矩阵（ 8×8 ）中右侧四列为横向上图像变化的高频成分，所以可以预测将其全部置零后，图片的横向灰度变化会放缓。而左侧四列为横向上图像变化的低频成分，且包含直流分量，可以预测将其全部置零后，图像基本只留下原图像中横向上黑白迅速切换的界限，也就是高频成分明显。针对该题，编写代码如下（即 `hw_2_3.m` 文件）：

```

1 clear all; close all; clc;
2
3 load("hall.mat");%读取图像信息
4
5 test=double(hall_gray(40:47,40:47));%取一测试块，8×8
6 imwrite(uint8(test),"hw_2_3_1.jpg");%输出原测试块图像
7 test=test-128*ones(8,8);
8 C=dct2(test);
9 C(1:8,5:8)=0;%DCT系数的右侧四列置零
10 test=idct2(C);
11 test=test+128*ones(8,8);
12 imwrite(uint8(test),"hw_2_3_2.jpg");%输出右侧四列置零后图像

```

```

13
14 test=double(hall_gray(40:47,40:47));
15 test=test-128*ones(8,8);
16 C=dct2(test);
17 C(1:8,1:4)=0;%DCT系数的左侧四列置零
18 test=idct2(C);
19 test=test+128*ones(8,8);
20 imwrite(uint8(test),"hw_2_3_3.jpg");%输出左侧四列置零后图像

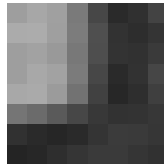
```

取出的 8×8 的测试图像块如下：



原测试图像块(hw_2_3_1.jpg)

其 DCT 系数右侧四列置零后图像如下：



DCT 系数右侧四列置零后图像(hw_2_3_2.jpg)

可以发现正如此前推测的那样，原来纵向的那条黑白界限的变化放缓了，也就是横向灰度变化缓慢了。其 DCT 系数左侧四列置零后图像如下：



DCT 系数左侧四列置零后图像(hw_2_3_3.jpg)

正如前面的推测那样，DCT 系数左侧四列置零后，图像基本只留下了原图中纵向的一道界限，也就是横向上的高频分量，同时纵向的高频分量可能也主要在 DCT 系数左侧四列的下方四行中得以体现，故而在 DCT 系数的左侧四列置零后，原图中那条横向的黑白界限也随之消失了。

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。

因为 $P = D^T C D$ ，对此进行转置后有 $P^T = D^T C^T D$ ，也就是说对 DCT 系数做转置其实和图片空域矩阵本身做转置没有区别。而将 DCT 系数（逆时针）旋转90度，就会导致一般系数较大的低频分量（矩阵左上角）被转移到一般系数较小的纵向高频分量（矩阵左下角）去，（逆时针旋转）图片将表现出纵向上明显的灰度快速变化（即出现横向条纹），而旋转180度，则会导致横向和纵向的耦合高频分量系数（矩阵右下角）明显增大，图片将在横向和纵向都表现出明显的灰度快速变化（即出现格状图像）。编写代码如下（即 hw_2_4.m 文件）：

```

1 clear all; close all; clc;
2
3 load("hall.mat");%读取图片信息
4
5 test=double(hall_gray(40:47,40:47));
6 test=test-128*ones(8,8);

```

```

7 C=dct2(test);
8 C=C.';
9 test=idct2(C);
10 test=test+128*ones(8,8);
11 imwrite(uint8(test),"hw_2_4_1.jpg");%输出DCT系数转置后的图像
12
13 test=double(hall_gray(40:47,40:47));
14 test=test-128*ones(8,8);
15 C=dct2(test);
16 C=rot90(C);
17 test=idct2(C);
18 test=test+128*ones(8,8);
19 imwrite(uint8(test),"hw_2_4_2.jpg");%输出DCT系数旋转90度后的图像
20
21 test=double(hall_gray(40:47,40:47));
22 test=test-128*ones(8,8);
23 C=dct2(test);
24 C=rot90(rot90(C));
25 test=idct2(C);
26 test=test+128*ones(8,8);
27 imwrite(uint8(test),"hw_2_4_3.jpg");%输出DCT系数旋转180度后的图像

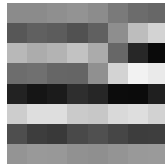
```

该测试图像块和上一题一致，对其进行转置后，图像如下：



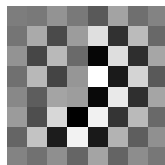
DCT 系数转置后图像(hw_2_4_1.jpg)

和前面结论一致，图像本身被转置了。将 DCT 系数旋转 90 度后图像如下：



DCT 系数逆时针旋转90度后图像(hw_2_4_2.jpg)

可以看到出现了明显的横向条纹，也就是纵向上灰度变化迅速，和前面结论一致。将 DCT 系数旋转 180 度后图像如下：



DCT 系数旋转180度后图像(hw_2_4_3.jpg)

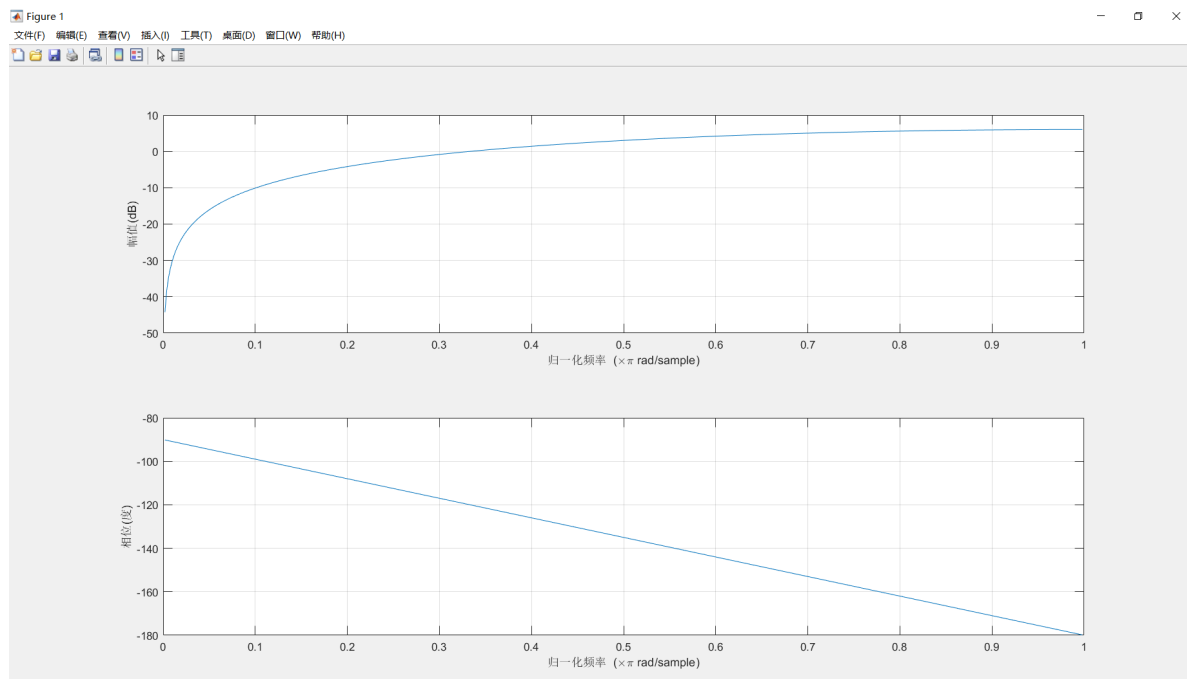
也和前面结论一致，图像变成了明显的格状，也就是在横向和纵向上灰度变化都十分迅速。

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。

编写如下代码（即 hw_2_5.m 文件）：


```
1 clear all; close all; clc;
2
3 freqz([-1,1],1);
```

得到差分编码系统的频率响应如下：



差分编码系统的频率响应

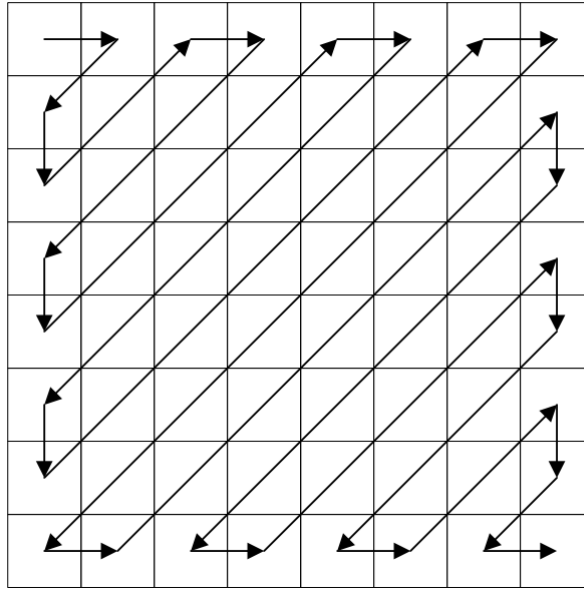
由上图可以看出，它是一个高通滤波器，说明 DC 系数的高频率分量更多。

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 ？

设 DC 预测误差取值为 x ，其对应的 Category 值为 c 。若 $x = 0$ ，则 $c = 0$ ；若 $x \neq 0$ ，则 $c = \lfloor \log_2(|x|) \rfloor + 1$ ，其中 $\lfloor \cdot \rfloor$ 符号表示向下取整。利用该式子就可以从预测误差推知其 Category。

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。

经过检索，我并没有找到很多实现 Zig-Zag 扫描的方法，如果从扫描速度方面来考量，那或许预先输入扫描顺序，然后直接读取矩阵中相应位置的元素的方法应该是最佳的，但是我认为这种方法有点笨拙，所以我总结了 Zig-Zag 扫描的特点，采用了另一种思路（无法确定是最佳的）实现了 Zig-Zag 扫描。



Zig-Zag 扫描示意图

由上面 Zig-Zag 扫描示意图可以发现，其实扫描就是不断在做向左下和向右上的两个方向的斜向运动，同时根据斜向运动起点的不同类别，也可以分成两种情况，第一种情况是对左上半区域的扫描，第二种情况是对右下半区域的扫描，只要知道是第几次斜向扫描，就可以明确判断出其扫描起始点和扫描方向。根据该思路，编写 Zig-Zag 扫描函数如下（即 myZigZag.m 文件）：

```

1  function Z=myZigZag(info)
2  %实现Zig-Zag扫描，将矩阵变换为向量
3
4  N=size(info,1);
5  Z=zeros(N^2,1);
6  num=0;
7
8  for i=1:2*N-1
9      if(mod(i,2)==0)
10         %扫描方向为从右上到左下
11         if(i<=N)
12             %左上半区域的扫描
13             for j=1:i
14                 num=num+1;
15                 Z(num,1)=info(j,i-j+1);
16             end
17         else
18             %右下半区域的扫描
19             for j=1:2*N-i
20                 num=num+1;
21                 Z(num,1)=info(i+j-N,N-j+1);
22             end
23         end
24     else
25         %扫描方向为从左下到右上
26         if(i<=N)
27             %左上半区域的扫描
28             for j=1:i
29                 num=num+1;
30                 Z(num,1)=info(i-j+1,j);
31             end
32         else

```

```

33         %右下区域的扫描
34         for j=1:2*N-i
35             num=num+1;
36             Z(num,1)=info(N-j+1,i+j-N);
37         end
38     end
39 end
40 end
41 end

```

再编写测试代码如下（即 hw_2_7.m 文件）：

```

1 clear all; close all; clc;
2
3 test=1:64;
4 test=reshape(test,8,8);%生成一从1到64的8×8矩阵
5 test=test.';
6 Z=myZigZag(test).' %输出扫描结果

```

输出结果如下：

```

Z =

列 1 至 20
    1     2     9    17    10     3     4    11    18    25    33    26    19    12     5     6    13    20    27    34

列 21 至 40
   41    49    42    35    28    21    14     7     8    15    22    29    36    43    50    57    58    51    44    37

列 41 至 60
   30    23    16    24    31    38    45    52    59    60    53    46    39    32    40    47    54    61    62    55

列 61 至 64
   48    56    63    64

```

Zig-Zag 扫描结果

可以看到成功实现了 Zig-Zag 扫描。

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，第一列为各个块的 DC 系数。

直接按照题目所给步骤，编写代码如下（即 hw_2_8.m 文件）：

```

1 clear all; close all; clc;
2
3 load("hall.mat");%读取图像信息
4 load("JpegCoeff.mat");%读取JPEG标准
5
6 xnum=size(hall_gray,1)/8;
7 ynum=size(hall_gray,2)/8;
8 info=double(hall_gray)-128*ones(8*xnum,8*ynum);%灰度值减去128
9 DCT_info=zeros(64,xnum*ynum);
10
11 %按照8×8的块依次进行DCT、量化，将其Zig-Zag扫描后存入DCT_info
12 for i=1:xnum
13     for j=1:ynum

```

```

14         temp=info((i-1)*8+1:i*8,(j-1)*8+1:j*8);
15         temp=dct2(temp);
16         temp=round(temp./QTAB);
17         temp=myZigZag(temp);
18         DCT_info(:,(i-1)*ynum+j)=temp;
19     end
20 end

```

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。

根据实验指导书中的讲解，（若图像长和宽不是 8 的倍数，则用 0 补齐）只要按照步骤可以很方便地实现 JPEG 编码：（在灰度值减去128后）首先依次计算各个块的 DCT 系数，并按照 JPEG 标准量化；此后进行 DC 编码，将 DC 系数进行差分编码后，利用前面提到的 DC 预测误差的取值和 Category 值之间的关系，算出 DC 值的 Category，然后找到 JPEG 标准表中 Category 对应的 Huffman 码，再在其后补上 DC 值的二进制表示（负数表示为 1 的补码），依次对每个 DC 值处理后即可完成 DC 编码；最后进行 AC 编码，AC 编码和 DC 编码很类似，只是增加了对前面零系数个数的统计，同时当出现连续 16 个零时插入 ZRL 码，在每个块的结尾插入 EOB 码。按照上述思路，编写代码如下（即 hw_2_9.m 文件）：

```

1  clear all; close all; clc;
2
3  load("hall.mat");%读取图像信息
4  load("JpegCoeff.mat");%读取JPEG标准
5
6  DCcode=[];%存储DC码流
7  ACcode=[];%存储AC码流
8
9  %补全为8的倍数
10 image=zeros(8*ceil(size(hall_gray,1)/8),8*ceil(size(hall_gray,2)/8));
11 image(1:size(hall_gray,1),1:size(hall_gray,2))=hall_gray;
12 image=uint8(image);
13
14 PicHeight=uint16(size(image,1));%存储图片高度
15 Picwidth=uint16(size(image,2));%存储图片宽度
16
17 %先计算DCT系数并量化
18 xnum=size(image,1)/8;
19 ynum=size(image,2)/8;
20 info=double(image)-128*ones(8*xnum,8*ynum);
21 DCT_info=zeros(64,xnum*ynum);
22
23 for i=1:xnum
24     for j=1:ynum
25         temp=info((i-1)*8+1:i*8,(j-1)*8+1:j*8);
26         temp=dct2(temp);
27         temp=round(temp./QTAB);
28         temp=myZigZag(temp);
29         DCT_info(:,(i-1)*ynum+j)=temp;
30     end
31 end
32
33 %对DC系数进行差分
34 Dctemp1=DCT_info(1,:);
35 Dctemp2=Dctemp1;

```

```

36 for i=2:size(DCTemp1,2)
37     DCTemp2(1,i)=DCTemp1(1,i-1)-DCTemp1(1,i);
38 end
39
40 %对DC系数进行编码
41 for i=1:size(DCTemp2,2)
42     if(DCTemp2(1,i)==0)
43         temp=[0,0];
44     else
45         temp=floor(log2(abs(DCTemp2(1,i))))+1;%计算相应Category
46         temp0=dec2bin(abs(DCTemp2(1,i)),temp)-'0';%计算二进制表示
47         if(DCTemp2(1,i)<0)%若为负数则取1的补码
48             temp0=~temp0;
49         end
50         temp=DCTAB(temp+1,2:DCTAB(temp+1,1)+1);
51         temp=[temp,temp0];
52     end
53     DCcode=[DCcode,temp];
54 end
55 DCcode=logical(DCcode);
56
57 %对AC系数进行编码
58 for i=1:size(DCT_info,2)
59     zeronum=0;%记录零系数个数
60     zrlnum=0;%记录ZRL个数
61     for j=2:64
62         if(DCT_info(j,i)==0)
63             zeronum=zeronum+1;
64             if(zeronum==16)
65                 zrlnum=zrlnum+1;
66                 zeronum=0;
67             end
68         else
69             tempx=[1,1,1,1,1,1,1,1,0,0,1];%ZRL码
70             temp=repmat(tempx,1,zrlnum);
71             ACcode=[ACcode,temp];%插入ZRL
72             zrlnum=0;
73
74             temp=floor(log2(abs(DCT_info(j,i))))+1;%计算Size
75             temp0=dec2bin(abs(DCT_info(j,i)),temp)-'0';%计算二进制表示
76             if(DCT_info(j,i)<0)%若为负数则取1的补码
77                 temp0=~temp0;
78             end
79             temp=ACTAB(zeronum*10+temp,4:ACTAB(zeronum*10+temp,3)+3);
80             temp=[temp,temp0];
81             zeronum=0;
82             ACcode=[ACcode,temp];
83         end
84     end
85     temp=[1,0,1,0];
86     ACcode=[ACcode,temp];%插入EOB
87 end
88 ACcode=logical(ACcode);
89
90 save jpegcodes DCcode ACcode PicHeight Picwidth %存储

```

10. 计算压缩比（输入文件长度/输出码流长度），注意转换为相同进制。

采用二进制表示，输入文件信息如下：

hall.mat (MAT 文件)	
名称	值
hall_color	120x168x3 uint8
hall_gray	120x168 uint8

输入文件信息

输入文件长度为 $120 \times 168 \times 8 = 161280$ ，而输出文件信息如下：

jpegcodes.mat (MAT 文件)	
名称	值
ACcode	1x23072 logical
DCcode	1x2031 logical
PicHeight	120
PicWidth	168

输出文件信息

输出文件长度为（高度和宽度按照 16 位来算） $23072 + 2031 + 16 + 16 = 25135$ 。故而压缩比为 $161280/25135 \approx 6.4166$ 。

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）和主观方法评价编解码效果如何。

对 JPEG 码流进行解码就是前面编码的逆过程，首先对 DC 码流解码和对 AC 码流解码：

- 对于 DC 码流解码，先一位一位读取码流，并对照 JPEG 标准中关于 DC 系数的 Huffman 码，如果吻合，则读取其对应 Category 位的后续码数，并将其由二进制转为十进制，DC 码流解码完成后再逆向进行差分编码即可得到 DC 系数；
- 对于 AC 码流解码，和 DC 码流基本一致（不需要逆向进行差分编码），只是出现 ZRL 码时连续插入 16 个零，而出现 EOB 码时，结束对当前块 AC 系数的解码，进入下一个块的解码。

此后再利用逆向 Zig-Zag 扫描，将各个块的数据向量恢复成矩阵，并利用量化矩阵，把这些矩阵复原成原本的 DCT 系数矩阵，最后把各个块的 DCT 系数进行 IDCT 运算，再组合成空域矩阵即可。按照上述思路，编写代码如下（即 hw_2_11.m 文件）：

```
1 clear all; close all; clc;
2
3 load("hall.mat");%读取原图像信息
4 load("jpegcodes.mat");%读取JPEG码流
5 load("JpegCoeff.mat");%读取JPEG标准
6
7 xnum=PicHeight/8;
8 ynum=PicWidth/8;
9 DCT_info=zeros(64,xnum*ynum);
10
11 %对DC码流进行解码
12 num=0;
13 i=1;
```

```

14 k=1;
15 cate=[];
16 while i<size(DCcode,2)
17     num=num+1;
18     if(num>=2)
19         cate=find(DCTAB==num);%查找相同长度Huffman码的位置
20     end
21     if(~isempty(cate))
22         for j=cate.'
23             if(all(DCTAB(j,2:1+num)==DCcode(1,i-num+1:i)))%比较是否一致
24                 if(num==2)
25                     DCT_info(1,k)=0;
26                 else
27                     DCT_info(1,k)=myBin2Dec(DCcode(1,i+1:i+j-1));
28                     i=i+j-1;
29                 end
30                 k=k+1;
31                 num=0;
32                 break;
33             end
34         end
35         cate=[];
36     end
37     i=i+1;
38 end
39
40 %DC系数差分编码的逆运算
41 for i=2:size(DCT_info,2)
42     DCT_info(1,i)=DCT_info(1,i-1)-DCT_info(1,i);
43 end
44
45 %对AC码流进行解码
46 k=1;%记录ACcode的位置
47 t=2;%记录是第几个AC系数
48 codelen=0;
49 siz=0;%AC编码中的size
50 zeronum=0;%存储前面零个数
51 ZRL=[1,1,1,1,1,1,1,1,0,0,1];%ZRL码
52 EOB=[1,0,1,0];%EOB码
53 for i=1:size(DCT_info,2)
54     while 1
55         codelen=codelen+1;
56         if(codelen==11 && all(ACcode(1,k-codelen+1:k)==ZRL))
57             codelen=0;
58             zeronum=zeronum+16;
59         else
60             pos=find(ACTAB(:,3)==codelen);
61             if(~isempty(pos))
62                 for j=pos.'
63                     if(all(ACTAB(j,4:3+codelen)==ACcode(1,k-codelen+1:k)))
64                         zeronum=zeronum+ACTAB(j,1);%跳过零
65                         siz=ACTAB(j,2);
66                         t=t+zeronum;
67                         DCT_info(t,i)=myBin2Dec(ACcode(1,k+1:k+siz));
68                         t=t+1;
69                         k=k+siz;
70                         zeronum=0;
71                         codelen=0;

```

```

72         break;
73     end
74     end
75     pos=[];
76     end
77     end
78     if(codeLen==4 && all(ACcode(1,k-codeLen+1:k)==EOB))%读取到结尾
79         k=k+1;
80         codeLen=0;
81         t=2;
82         zeronum=0;
83         break;
84     end
85     k=k+1;
86 end
87 end
88
89 %进一步恢复出图片信息
90 for i=1:xnum
91     for j=1:ynum
92         temp=DCT_info(:,(i-1)*ynum+j);
93         temp=myReverseZigZag(temp);
94         temp=temp.*QTAB;
95         temp=idct2(temp);
96         info((i-1)*8+1:i*8,(j-1)*8+1:j*8)=temp;
97     end
98 end
99 info=info+128*ones(8*xnum,8*ynum);
100
101 %计算PSNR
102 MSE=0;
103 for i=1:PicHeight
104     for j=1:Picwidth
105         MSE=MSE+(double(hall_gray(i,j))-double(info(i,j)))^2;
106     end
107 end
108 MSE=MSE/(double(Picwidth)*double(PicHeight));
109 PSNR=10*log10(255^2/MSE) %输出PSNR
110
111 imwrite(uint8(hall_gray),"hw_2_11_1.jpg");%输出原图像
112 imwrite(uint8(info),"hw_2_11_2.jpg");%输出解码出的图像

```

计算得到 PSNR 值如下：

```

命令行窗口

PSNR =

    31.1771

fx >>

```

PSNR 值

原图像如下：



原图像(hw_2_11_1.jpg)

最后解码出的图像如下：



解码出的图像(hw_2_11_2.jpg)

从主观感受上来说，编解码出的图像和原图像基本一致，只有仔细看才能看出稍变模糊了一些（比如云朵边缘），这反映出 JPEG 编码的强大，在实现如此大压缩比的同时，图片质量依旧有保证。

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

为了方便，我把 JPEG 编码和解码打包为两个函数 myJPEGencode 和 myJPEGdecode，编写代码如下（即 hw_2_12.m 文件）：

```
1 clear all; close all; clc;
2
3 load("hall.mat");%读取原图像信息
4
5 %量化步长减半的JPEG编码
6 [DCcode,ACcode,PicHeight,Picwidth]=myJPEGencode(hall_gray,0.5);
7
8 save jpegcodes2 DCcode ACcode PicHeight Picwidth %存储
9
10 %量化步长减半的JPEG解码
11 info=myJPEGdecode(DCcode,ACcode,PicHeight,Picwidth,0.5);
12
13 %计算PSNR
14 MSE=0;
15 for i=1:PicHeight
```

```

16     for j=1:Picwidth
17         MSE=MSE+(double(hall_gray(i,j))-double(info(i,j)))^2;
18     end
19 end
20 MSE=MSE/(double(Picwidth)*double(PicHeight));
21 PSNR=10*log10(255^2/MSE) %输出PSNR
22
23 imwrite(info,"hw_2_12_1.jpg");%输出解码后图像

```

量化步长减半后的输出文件信息如下：

jpegcodes2.mat (MAT 文件)		
<input type="checkbox"/>	名称	值
<input checked="" type="checkbox"/>	ACcode	1x34164 logical
<input checked="" type="checkbox"/>	DCcode	1x2410 logical
<input type="checkbox"/>	PicHeight	120
<input type="checkbox"/>	PicWidth	168

量化步长减半后的输出文件信息

计算得输出文件码流长度为 $34164 + 2410 + 16 + 16 = 36606$ ，则压缩比为 $161280/36606 \approx 4.4058$ ，压缩比相较于前面量化步长未减半的编码有所下降。PSNR 值如下：

```

命令行窗口

PSNR =

    34.2067

fx >>

```

量化步长减半后的 PSNR 值

可以看到，量化步长减半后 PSNR 值有所提升，也就是图像质量有所提高。量化步长减半后，编解码出的图像如下：



量化步长减半后解码出的图像(hw_2_12_1.jpg)


对比前面的解码出的图像，确实更清晰了一点。

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。

编写代码如下（即 hw_2_13.m 文件）：





```
1 clear all; close all; clc;
2
3 load("snow.mat");%读取原图像信息
4
5 imwrite(uint8(snow),"hw_2_13_1.jpg");%输出原雪花图像
6
7 [DCcode,ACcode,PicHeight,PicWidth]=myJPEGencode(snow,1);%编码
8
9 save jpegcodes3 DCcode ACcode PicHeight PicWidth %存储
10
11 info=myJPEGdecode(DCcode,ACcode,PicHeight,PicWidth,1);%解码
12
13 imwrite(info,"hw_2_13_2.jpg");%输出解码后图像
14
15 %计算PSNR
16 MSE=0;
17 for i=1:PicHeight
18     for j=1:PicWidth
19         MSE=MSE+(double(snow(i,j))-double(info(i,j)))^2;
20     end
21 end
22 MSE=MSE/(double(PicWidth)*double(PicHeight));
23 PSNR=10*log10(255^2/MSE) %输出PSNR
```

输入的雪花图像信息如下：

snow.mat (MAT 文件)		
	名称	值
	snow	128x160 uint8

输入文件信息

输入文件长度为 $128 \times 160 \times 8 = 163840$ ，而输入的文件编码后输出文件信息如下：

jpegcodes3.mat (MAT 文件)		
	名称	值
	ACcode	1x43546 logical
	DCcode	1x1403 logical
	PicHeight	128
	PicWidth	160

输出文件信息

输出文件码流长度为 $43546 + 1403 + 16 + 16 = 44981$ ，由此得压缩比为 $163840/44981 \approx 3.6424$ 。编解码后图像相对于原图像的 PSNR 值如下：

```
命令行窗口

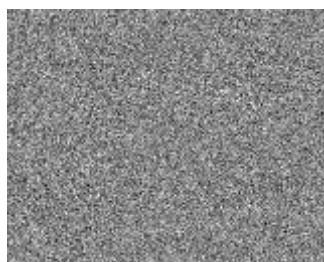
PSNR =

    22.9244

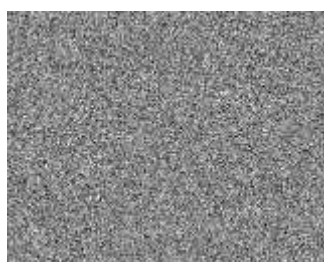
fx >>
```

PSNR 值

原雪花图像和编解码后雪花图像分别如下：



原雪花图像(hw_2_13_1.jpg)



编解码后图像(hw_2_13_2.jpg)

可以看到雪花图像编解码的压缩比和 PSNR 值（以及直观图像质量）相较于测试图像都有很大明显降低，这是因为雪花图像中含有很多的高频交流成分，这样就导致在编码时 AC 码流很长，因此压缩比较低，同时正因为雪花图像的 DCT 系数矩阵中很多高频交流分量不为零，因此进行量化的数据量就更大，而量化进行地越多，自然噪声就容易增多，故而 PSNR 值也较低。

第三章练习题

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。

空域隐藏方法比较简单，直接将数据隐藏在各个像素块的最低比特位即可，编写代码如下（即 hw_3_1.m 文件）：

```
1 clear all; close all; clc;
2
3 load("hall.mat");%读取图像信息
4
5 info=hall_gray;
6 msg=zeros(1,size(hall_gray,1)*size(hall_gray,2));%存储要发送的信息
7
8 %初始化信息
9 for i=1:size(msg,2)
10     msg(1,i)=randi([0,1]);
```

```

11 end
12
13 %将信息存入像素灰度值比特最低位
14 for i=1:size(info,1)
15     for j=1:size(info,2)
16         info(i,j)=info(i,j)-mod(info(i,j),2)+msg(1,(i-1)*size(info,2)+j);
17     end
18 end
19
20 imwrite(hall_gray,"hw_3_1_1.jpg");%输出插入信息前的原图像
21 imwrite(info,"hw_3_1_2.jpg");%输出插入信息后的图像
22 [DCcode,ACcode,PicHeight,PicWidth]=myJPEGencode(info,1);%编码
23
24 decode_info=myJPEGdecode(DCcode,ACcode,PicHeight,PicWidth,1);%解码
25
26 %验证解码出的信息的正确比例
27 correct=0;
28 for i=1:size(info,1)
29     for j=1:size(info,2)
30         if(bitget(decode_info(i,j),1)==msg(1,(i-1)*size(info,2)+j))
31             correct=correct+1;
32         end
33     end
34 end
35 correct_percentage=correct/size(msg,2) %输出正确比例

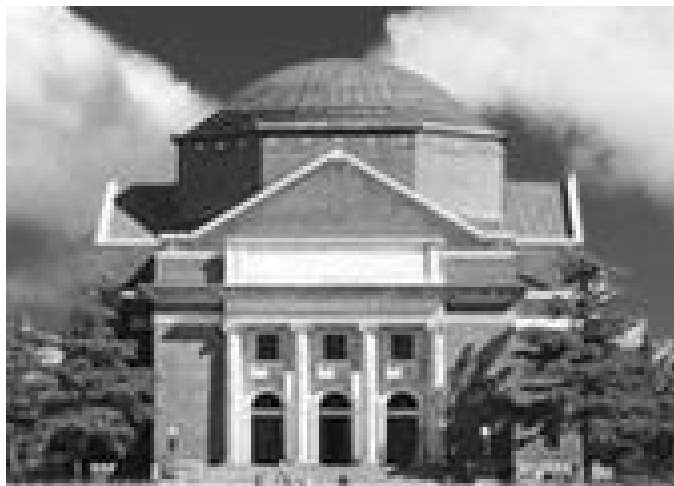
```

(插入的信息是随机生成的) 插入信息前的原图像如下:



插入信息前的原图像(hw_3_1_1.jpg)

插入信息后的图像如下:



插入信息后的图像(hw_3_1_2.jpg)

可以看到因为插入的是最低比特位，对图像质量几乎没有影响。该空域隐藏信息方法经过 JPEG 编解码后，再解析出信息，和原信息对比，正确率如下：

```
命令行窗口

correct_percentage =

    0.5010

fx >>
```

JPEG 编解码后解读出信息的正确率

可以看到该空域隐藏信息的方法抗 JPEG 编码能力很差，经过 JPEG 编解码后正确率基本只能维持在 50% 左右。

2. 依次实现本章介绍的三种变换域信息隐藏方法和提取方法，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。

本章介绍的三种变换域信息隐藏方法分别为：

- 用信息逐一替换量化后 DCT 系数最低位，再进行熵编码；
- 用信息逐一替换若干量化后的 DCT 系数最低位，不是每个 DCT 系数都嵌入信息，再进行熵编码；
- 将待隐藏信息用 -1, 1 序列表示后，逐一追加在每块 Zig-Zag 序列最后一个非零的 DCT 系数之后，若无非零系数则替换最后一个系数，再进行熵编码。

按照上述思路，编写代码如下（即 hw_3_2.m 文件）：

```
1 clear all; close all; clc;
2
3 load("hall.mat");%读取图像信息
4 load("JpegCoeff.mat");%读取JPEG标准
5
6 info=hall_gray;
7 PicHeight=size(info,1);
8 Picwidth=size(info,2);
9 msg=zeros(1,PicHeight*Picwidth);%存储需要发送的信息
10
11 %初始化信息
12 for i=1:size(msg,2)
```

```

13     msg(1,i)=randi([0,1]);
14 end
15
16 %计算DCT系数
17 xnum=size(info,1)/8;
18 ynum=size(info,2)/8;
19 tempinfo=double(info)-128*ones(8*xnum,8*ynum);
20 DCT_info=zeros(64,xnum*ynum);
21
22 %量化并存储
23 for i=1:xnum
24     for j=1:ynum
25         temp=tempinfo((i-1)*8+1:i*8,(j-1)*8+1:j*8);
26         temp=dct2(temp);
27         temp=round(temp./QTAB);
28         temp=myZigZag(temp);
29         DCT_info(:,(i-1)*ynum+j)=temp;
30     end
31 end
32
33 %第一种DCT域信息隐藏方法
34 DCT_info1=DCT_info;
35 for i=1:size(DCT_info,1)
36     for j=1:size(DCT_info,2)
37         DCT_info1(i,j)=DCT_info(i,j)-...
38             mod(DCT_info(i,j),2)+msg(1,(i-1)*size(DCT_info,2)+j);
39     end
40 end
41 [DCcode1,ACcode1]=myJPEGencode2(DCT_info1);%编码
42 [info1,deDCT_info1]=myJPEGdecode2(DCcode1,ACcode1,PicHeight,Picwidth);%解码
43
44 %验证解码出的信息的正确比例
45 correct=0;
46 for i=1:size(deDCT_info1,1)
47     for j=1:size(deDCT_info1,2)
48         if(mod(deDCT_info1(i,j),2)==msg(1,(i-1)*size(deDCT_info1,2)+j))
49             correct=correct+1;
50         end
51     end
52 end
53 correct_percentage1=correct/size(msg,2) %输出正确比例
54
55 compress_ratio1=PicHeight*Picwidth*8/...
56 (32+size(ACcode1,2)+size(DCcode1,2)) %输出压缩比
57
58 PSNR1=myPSNR(hall_gray,info1) %输出PSNR值
59
60 imwrite(info1,"hw_3_2_1.jpg");%输出第一种方法处理后图像
61
62 %第二种DCT域信息隐藏方法，把信息存在各块DCT矩阵中的靠上4行中
63 DCT_info2=DCT_info;
64 for i=1:32
65     for j=1:size(DCT_info,2)
66         DCT_info2(i,j)=DCT_info(i,j)-...
67             mod(DCT_info(i,j),2)+msg(1,(i-1)*size(DCT_info,2)+j);
68     end
69 end
70 [DCcode2,ACcode2]=myJPEGencode2(DCT_info2);%编码

```

```

71 [info2,deDCT_info2]=myJPEGdecode2(DCcode2,ACcode2,PicHeight,Picwidth);%解码
72
73 %验证解码出的信息的正确比例
74 correct=0;
75 for i=1:32
76     for j=1:size(deDCT_info2,2)
77         if(mod(deDCT_info2(i,j),2)==msg(1,(i-1)*size(deDCT_info2,2)+j))
78             correct=correct+1;
79         end
80     end
81 end
82 correct_percentage2=correct/(32*size(deDCT_info2,2)) %输出正确比例
83
84 compress_ratio2=PicHeight*Picwidth*8/...
85 (32+size(ACcode2,2)+size(DCcode2,2)) %输出压缩比
86
87 PSNR2=myPSNR(hall_gray,info2) %输出PSNR值
88
89 imwrite(info2,"hw_3_2_2.jpg");%输出第二种方法处理后图像
90
91 %第三种DCT域信息隐藏方法
92 DCT_info3=DCT_info;
93 for j=1:size(DCT_info,2)
94     pos=find(DCT_info(:,j)~=0);%搜索最后一个非零系数位置
95     if(max(pos)==64)
96         pos=63;
97     else
98         pos=max(pos);
99     end
100     if(msg(j)==0)
101         DCT_info3(pos+1,j)=-1;
102     else
103         DCT_info3(pos+1,j)=1;
104     end
105 end
106 [DCcode3,ACcode3]=myJPEGencode2(DCT_info3);%编码
107 [info3,deDCT_info3]=myJPEGdecode2(DCcode3,ACcode3,PicHeight,Picwidth);%解码
108
109 %验证解码出的信息的正确比例
110 correct=0;
111 for i=1:size(deDCT_info3,2)
112     pos=find(deDCT_info3(:,i)~=0);%搜索最后一个非零系数位置
113     pos=max(pos);
114     if(deDCT_info3(pos,i)==1 && msg(1,i)==1)
115         correct=correct+1;
116     elseif(deDCT_info3(pos,i)==-1 && msg(1,i)==0)
117         correct=correct+1;
118     end
119 end
120 correct_percentage3=correct/size(deDCT_info3,2) %输出正确比例
121
122 compress_ratio3=PicHeight*Picwidth*8/...
123 (32+size(ACcode3,2)+size(DCcode3,2)) %输出压缩比
124
125 PSNR3=myPSNR(hall_gray,info3) %输出PSNR值
126
127 imwrite(info3,"hw_3_2_3.jpg");%输出第三种方法处理后图像

```


(插入的信息是随机生成的) 第一种方法的正确率、压缩比和 PSNR 值如下:

命令行窗口

```
correct_percentagel =  
  
1  
  
compress_ratio1 =  
  
2.8539  
  
PSNR1 =  
  
15.3943
```

第一种变换域嵌密方法的正确率、压缩比和 PSNR 值

第二种方法的正确率、压缩比和 PSNR 值如下:

命令行窗口

```
correct_percentage2 =  
  
1  
  
compress_ratio2 =  
  
4.6733  
  
PSNR2 =  
  
23.2593
```

第二种变换域嵌密方法的正确率、压缩比和 PSNR 值

第三种方法的正确率、压缩比和 PSNR 值如下:

命令行窗口

```
correct_percentage3 =  
  
1  
  
compress_ratio3 =  
  
6.1840  
  
PSNR3 =  
  
29.0321
```

第三种变换域嵌密方法的正确率、压缩比和 PSNR 值

而三种方法嵌密后的图像分别如下：



第一种方法嵌密后图像(hw_3_2_1.jpg)



第二种方法嵌密后图像(hw_3_2_2.jpg)



第三种方法嵌密后图像(hw_3_2_3.jpg)

和未嵌密的图像相对比，可以发现经过这三种方法嵌密后的图像质量（从主观感受上来看以及从 PSNR 值上来看）以及压缩比都有所下降（第二种方法把信息存储在了每个块的 DCT 系数矩阵的前四行的最低比特位，此时第三种方法影响最小，第二种方法次之，第一种方法最末），因为三种方法都从不同程度上引入了一些原本不存在的高频分量，所以导致编码后码流变长，同时噪声也难以避免的增多。

相较于空域嵌密方法，这些变换域嵌密方法的隐蔽性或许更差一些（在本次测试中，第三种方法隐蔽性>第二种方法隐蔽性>第一种方法隐蔽性），因为从直观感受来讲，图片质量有较为明显的下降，但是这种嵌密方法不容易被直接破解，同时这些方法抗 JPEG 编码的能力极强，都能保证 JPEG 编解码后信息正确率为 100%，因为插入信息的过程处于 JPEG 编解码过程中往往不会产生误差的环节。

如果比较这三种方法的优劣，第二和第三种方法对图像质量和压缩比的影响往往小于第一种方法，此外测试发现在使用第二种方法时如果选取插入信息的 DCT 系数位置合理（比如把信息插入 DC 系数中），第二种方法优于第三种方法，当然这种比较仅仅基于图像质量和压缩比的变化，如果要考虑到存储信息的多少，则可能相对优劣的比较结果又有所不同。

第四章练习题

1. 所给资料 Faces 目录下包含从网图中截取的 28 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，是否需要首先将图像调整为相同大小？

(b) 假设 L 分别取 3, 4, 5，所得三个 v 之间有何关系？

用于判断图像相近度的标准本质上是图像中各种颜色的比例，而颜色比例一般不会因为图像大小而改变，所以不需要将图像调整为相同大小。

L 分别取 3, 4, 5 时，其对应 v 的关系为：

- $$v_3(2^6 \times r + 2^3 \times b + g + 1) = \sum_{x,y,z=0,1} v_4(2^8 \times (2r + x) + 2^4 \times (2b + y) + (2g + z) + 1)$$
- $$v_4(2^8 \times R + 2^4 \times B + G + 1) = \sum_{x,y,z=0,1} v_5(2^{10} \times (2R + x) + 2^5 \times (2B + y) + (2G + z) + 1)$$

其中 r, b, g 的范围为 $[0, 2^3 - 1]$ ， R, B, G 的范围为 $[0, 2^4 - 1]$ ，其与图片的像素点的 RGB 值相对应， v_n 表示 L 取 n 时训练出的 v ， $v_n(m)$ 的含义是训练出的 v 向量的第 m 个元素值。

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，评价检测结果有何区别。

检测人脸的核心思路和实验指导书中一致，就是以图像中各个颜色的比例向量作为其特征，先用大量样本训练出标准向量 v （取各个样本颜色比例向量的平均值），然后定义某种距离算法，当一块图像的颜色比例向量 u 和标准向量 v 距离足够近时，认定其为人脸。

不过实验指导书并没有给出如何确定人脸图框的大小，我采用直接暴力搜索的方法，即首先定义一个图框大小初值（参考图片本身大小），然后在被检测图像中连续移动该图框，计算图框中图像的颜色比例向量并和标准向量 v 进行距离比较，每完成一轮检测就改变一点图框大小，直到图框小到某一极限，如果在这一过程中检测到人脸（即图框中图像的颜色比例向量 u 和 v 距离够近），则清除该人脸，重新开始新一轮检测，如果一轮完整的检测没有发现人脸，则检测结束。

在初次尝试时，发现不同人脸的颜色比例向量与标准向量 v 的距离的跨度较大，仅定义一个固定的距离阈值容易出现误判和漏判，所以我又对检测流程进行了一点修改：当某一轮完整检测（图框由最大到最小）没有发现人脸，不立即停止检测，而是适度放宽距离阈值，继续检测，直到未发现人脸的次数达到某一设定值。这样修改之后，检测效果有了明显提升。按照上述思路，编写代码如下：

```
1 clear all; close all; clc;
2
3 image=imread("test.jpg");%读取检测人脸的图像信息
4 test=image;%暂存图像信息
5
6 L=5;%确定RGB值的比特位数
7
8 v=myTraining(L);%利用样本进行训练得到v
9
10 %修改以下参数来改善检测结果
11 ratioh=5;%人脸在图片高度所占最大比例的倒数
12 ratiow=5;%人脸在图片宽度所占最大比例的倒数
13 e=0.63;%距离阈值
14 num=20;%人脸占图比例最小值
15 step=10;%每次移动的像素数
16 ratio=1.8;%人脸图框比例极限
17 deepnum=1;%再进一步深入检测次数
18 loose=0.05;%深入检测时放宽的距离阈值
19
20 h1=floor(size(test,1)/ratioh);%高度迈进步长初始值
21 w1=floor(size(test,2)/ratiow);%宽度迈进步长初始值
22 h2=floor(size(test,1)/num);%高度迈进步长终止值
23 w2=floor(size(test,2)/num);%宽度迈进步长终止值
24
25 facepos=[];%记录人脸位置和大小信息
26 flag=0;%标志是否检测到了人脸
27
28 deep=0;%深入检测次数
29 facenum=0;%检测到的人脸数
30
31 while 1
32     flag=0;
33     for h=h1:-step:h2
34         for w=w1:-step:w2
35             if(w/h<1/ratio)
36                 break;
37             end
38             if(w/h>ratio)
```

```

39         continue;
40     end
41     i=1;
42     while i+h-1<=size(test,1)
43         j=1;
44         while j+w-1<=size(test,2)
45             u=myCalculate(test(i:i+h-1,j:j+w-1,:),L);
46             if(myDistance(u,v)<e)
47                 flag=1;
48                 facepos=[facepos;i,j,h,w];%存储已检测到的人脸信息
49                 test(i:i+h-1,j:j+w-1,1:3)=0;%清除已检测到的人脸
50                 break;
51             end
52             j=j+w;
53         end
54         i=i+h;
55         if(flag==1)
56             break;
57         end
58     end
59     if(flag==1)
60         break;
61     end
62 end
63 if(flag==1)
64     facenum=facenum+1 %输出当前检测到的人脸数量
65     break;
66 end
67 end
68 %若需要加快检测速度，可以加上下面的代码，限制图框大小变化范围
69 %     if(facenum==1)
70 %         h1=h+4*step;
71 %         w1=w+4*step;
72 %     end
73 if(flag==0)
74     if(deep==deepnum)%放宽距离阈值，进一步检测
75         break;
76     else
77         e=e+loose;
78         deep=deep+1;
79     end
80 end
81 end
82
83 image=myDrawFrame(facepos,image);%画上红色图框
84 imwrite(image,"hw_4_2_1.jpg");%输出检测结果

```

为了提升检测速度，避免耗费时间的盲目搜索，我在代码中对图框的比例以及其大小变化范围作了限定，将这些限定与距离阈值、深度搜索次数以及每轮距离阈值宽限值根据图像特点做适当调整，可以使检测结果更佳。

输入的测试图像如下：



测试图像(test.jpg)

当 $L = 5$, 距离初始阈值为 0.63 (其他参数和展示代码中一致) , 检测结果如下:



L 取 5 时的检测结果(hw_4_2_1.jpg)

当 $L = 4$, 距离初始阈值为 0.52 (其他参数和展示代码中一致) , 检测结果如下:



L 取 4 时的检测结果(hw_4_2_2.jpg)

当 $L = 3$, 距离初始阈值为 0.45 (其他参数和展示代码中一致) , 检测结果如下:



L 取 3 时的检测结果(hw_4_2_3.jpg)

可以发现，当 L 取值越小时，为了达到更好的检测效果，初始距离阈值需要适当的减小。由于采用了多轮逐渐放宽距离阈值进一步检测的方法，一些肤色相较于标准向量差别较大的人脸也被识别到了，在全部人脸被识别到的同时没有产生误判，且大部分人脸还都位于图框中心，所以该算法还是比较成功的。

3. 对上述图像分别如下处理后

- (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。

由于此题代码和上一题相比基本没有太多变动，故不再展示，三个小问分别对应于 hw_4_3_1.m、hw_4_3_2.m、hw_4_3_3.m 文件。

在图像顺时针旋转 90° 后（参数和上题中 L 取 5 时的参数一致），人脸检测结果如下：



图像顺时针旋转 90° 后的检测结果(hw_4_3_1.jpg)

可以看到效果还是很不错的，因为旋转不会影响图像块中颜色比例，所以人脸都被检测到了，且没有误判，只是一些框的比例和人脸不是很贴合，同时小部分框的中心略微偏离人脸，这和距离阈值的选取有关（距离阈值选的偏大了），或许还和我对图框大小、比例的调整策略有关，我代码中在改变图框大小时，会先减小图框的宽，遍历不同宽度后，再减小图框的高，因为一般人脸都是上下长度大于左右宽度的，但是旋转后，恰好就反过来了。

在图像宽度拉伸为原来 2 倍后（参数和上题中 L 取 5 时的参数一致），人脸检测结果如下：



图像宽度拉伸为原来 2 倍后的检测结果(hw_4_3_2.jpg)

可以看到检测效果还是很不错的，同样是因为图像拉伸不会改变图框中图像的颜色比例，所以人脸都被检测到了且没有误判，但是仍然存在部分图框大小和人脸大小不是很贴合以及一些图框中心略微偏离人脸的情况，原因和前面提到的一致。

适当改变图像颜色后（深入检测次数改为 6，其他参数和上题中 L 取 5 时的参数一致），人脸检测结果如下：



适当改变图像颜色后的检测结果(hw_4_3_3.jpg)

可以看到即使深入检测次数改为了 6（此时最后一轮检测的距离阈值已经为 0.93），也只是零星地检测到了几个人脸（其中还存在把几个人脸框成一个的情况），有不少的漏判，这是因为算法本身就是基于图像颜色的，颜色变化了，自然很难检测出人脸。

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？

我觉得如果要重新选择人脸样本，最好选取和需要检测的人脸肤色相近、姿态相同（比如都是正脸）、无佩戴物且全部为人脸无背景的图片，这样效果应该最佳。（如果训练样本中包含多种肤色差距较大的人种，或许训练出的标准更普适，但是在用于检索特定肤色人种的人脸时可能效果反而不如仅包含该人种人脸的样本）

总结

图像处理大作业在我看来十分有趣，在进一步熟悉 matlab 编程的同时，还了解了 JPEG 编解码、图片嵌密以及基于颜色的人脸识别。在完成 JPEG 编解码以及图片嵌密部分时都十分顺利，但是在人脸识别部分，受限于自身知识储备，我感觉自己基于颜色实现的人脸识别还有很多可改进之处：

- 采用暴力算法遍历图框大小，这样检测时间较长，感觉可以利用二叉树进一步优化，或者可以把第一次搜索到人脸的大小信息利用起来，进一步限制图框大小遍历的范围（该方法我已做了初步尝

试，放在了代码注释中)；

- 偶尔会出现图框偏大或者略微偏离人脸的现象，我觉得可以在检测到一个人脸后，设计算法再进一步缩小图框大小并细调其位置，使得定位更精准。