

Matlab 高级编程与工程应用

图像处理大作业

姓名：王浩宇

班级：无 82

学号：2018010389

目录

第一部分：基础知识 2

第二部分：图像压缩编码 6

第三部分：信息隐藏 35

第四部分：人脸检测 50

基础知识:

1.1 在命令窗口输入 `help images` 可以查看该工具箱内的所有函数，阅读后大致了解函数的功能

(1) 主要思想:

在命令行中输入 `help images` 即可获得图像处理工具箱中的函数，阅读函数说明页即可获得函数的功能。

(2) 运行结果:

通过函数参考页了解本实验中常用函数的基本功能。例如:

保存图像函数 `imwrite` 函数可以将 `uint8` 类型的图像生成 `jpg`、`bmp` 等文件存入指定的地址。

`imshow` 函数可以显示 `uint8` 类型的图像。

`dct2` 函数可以对图像进行相应 `dct` 变换。

1.2 利用 matlab 提供的 image file I/O 函数完成以下处理

(a) 以测试图像中心为圆心，图像长和宽中较小值的一半为半径画一个红色的圆

(b) 将测试图像涂成国际象棋的“黑白棋”状，黑代表黑色，白代表保留原图，使用同一个看图软件浏览上述两图，检测是否达标

(1) 主要思想:

画圆时，先获取待处理图像的 `size` 参数，确定图像的中心，使用 `for` 循环对每一个像素点进行扫描。如果当前像素点到图像中心的距离小于图像最短边的一半时，则将图像的 `R` 分量设置为 255，其余两个分量设置为 0(第二种设置方式为将 `R` 分量设置为 255，其余分量保持不变。)即可得到画好红色圆的图像。在绘制网格状图像时，需要判断当前像素点所处的位置，如果处在白区则保持各个分量不变。如果处在黑区则将所有分量均设置为 0。

(2) 核心代码:

以测试图像的中心画圆

```
clear all;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat
hall_color = double(hall_color);
r = min(size(hall_gray))/2;
dim = size(hall_color);
for i = 1 : 1 : dim(1)
    for j = 1 : 1 : dim(2)
        if (i-dim(1)/2)^2 + (j-dim(2)/2)^2 < r^2
            hall_color(i,j,1) = 255;
            hall_color(i,j,2) = 0;           %第二种图像将该行注释掉
            hall_color(i,j,3) = 0;           %第二种图像将该行注释掉
        else
            end
    end
end
imwrite(uint8(hall_color),'C:\\Users\\HJSF\\Desktop\\Dynamic\\hall_v2.jpg');
```

将测试图像设置为国际象棋黑白网格状:

```
clear all;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat
hall_color = double(hall_color);
flag = hall_color;
dim = size(hall_color);
len = 8;
for i = 1 : 1 : dim(1)
    for j = 1 : 1 : dim(2)
        hall_color(i,j,1) =
hall_color(i,j,1)*mod(floor(i/len)+floor(j/len),2);
        hall_color(i,j,2) =
hall_color(i,j,2)*mod(floor(i/len)+floor(j/len),2);
        hall_color(i,j,3) =
hall_color(i,j,3)*mod(floor(i/len)+floor(j/len),2);
    end
end
```

```
imwrite(uint8(hall_color), 'C:\\Users\\HJSF\\Desktop\\Dynamic\\hall_ch  
essmap.jpg');
```

(3) 核心代码说明:

在画圆部分中, 如果想要得到第一种画圆图像则将 R 分量设置为 255, 其余分量设置为 0。如果想得到第二种画圆图像, 则将 R 分量设置为 255, 其余分量保持不变。

在画网格图的部分中设置网格的大小 8×8 像素, 使用行序号与列序号记录当前网格所在的位置。如果行序号与列序号之和为奇数则位于白区, 若序号之和为偶数, 则位于黑区。

(4) 运行结果:

以测试图像为中心做出的红色的圆如下:



version1



version2

将原始图像加上网格图:



(5) 结果分析:

如上图所示, 第一种画圆方式在圆内全是纯红色, 没有其他的颜色。第二种画圆方式将圆内的能感受到颜色十分红, 同时也能看出圆内的其他颜色。加上网格图后, 可以看出处理后的图像是网格相间的, 满足题目要求。因为图像的尺寸为 120×168 , 因此处理后的图像网格应该是 15×21 , 满足设计要求。

图像压缩编码：

2.1 图像的预处理是将每个像素的灰度值减去 128，这个步骤能否在变换域中进行？请在测试图像中截取一块验证结论。

(1) 主要思想：

经过数学推导验证，可以在变换域上实现将每个像素的灰度值减去 128。具体的做法为，对于一个 8×8 的矩阵块，先对该矩阵进行 DCT 变换得到 DCT 系数矩阵，将该 DCT 系数矩阵减去 $128 \times \text{ones}(8,8)$ 的 DCT 系数，差矩阵经过逆变换得到的矩阵即为原始矩阵减去 128。

(2) 核心代码：

测试代码：

```
clear all, close all, clc;
load hall.mat
image_test = zeros(8,8);
for i = 1 : 1 : 8
    for j = 1 : 1 : 8
        image_test(i,j) = hall_gray(i,j);
    end
end
MatC = DCT(image_test);
AC = MatC - DCT(128*ones(8,8));
pre_image = IDCT(AC);
dif = image_test - pre_image - 128;
```

DCT 函数：

```
function MatC = DCT(MatP)
% get 2d dct transform
% MatP is a square matrix
sizeP = size(MatP);
if sizeP(1) ~= sizeP(2)
    disp("Error! MatP is not a square matrix!");
    return;
end
dim = length(MatP);
MatD = zeros(dim,dim);
for i = 1 : 1 : dim
    if i == 1
```

```

        MatD(i,:) = sqrt(1/2)*ones(1,dim);
    else
        MatD(i,:) = cos(pi/(2*dim)*(i-1:2*(i-1):(i-1)*(2*dim-1)));
    end
end
MatC = 2/dim*MatD*MatP*(MatD. ');
end

```

IDCT 函数:

```

function MatP = IDCT(MatC)
    sizeC = size(MatC);
    if sizeC(1) ~= sizeC(2)
        disp("Error! MatC is not a square matrix!");
        return;
    end
    dim = length(MatC);
    MatD = zeros(dim,dim);
    for i = 1 : 1 : dim
        if i == 1
            MatD(i,:) = sqrt(1/2)*ones(1,dim);
        else
            MatD(i,:) = cos(pi/(2*dim)*(i-1:2*(i-1):(i-1)*(2*dim-1)));
        end
    end
    MatP = 2/dim*(MatD. ') * MatC * MatD;
end

```

(3) 运行结果:

得到的 dif 差矩阵为

变量 - dif									
pre_image x dif x									
8x8 double									
	1	2	3	4	5	6	7	8	
1	-9.9476e-...	-1.4211e-...	5.6843e-14	-4.2633e-...	0	8.5265e-14	-8.5265e-...	5.6843e-14	
2	-5.6843e-...	2.8422e-14	1.1369e-13	-2.8422e-...	5.6843e-14	8.5265e-14	-5.6843e-...	8.5265e-14	
3	-2.8422e-...	5.6843e-14	5.6843e-14	0	5.6843e-14	1.1369e-13	-2.8422e-...	5.6843e-14	
4	-1.7053e-...	-8.5265e-...	-4.2633e-...	-9.9476e-...	-4.2633e-...	0	-1.7053e-...	2.8422e-14	
5	-8.5265e-...	2.8422e-14	2.8422e-14	0	2.8422e-14	1.1369e-13	-5.6843e-...	5.6843e-14	
6	-7.1054e-...	5.6843e-14	5.6843e-14	2.8422e-14	5.6843e-14	1.1369e-13	-8.5265e-...	1.4211e-13	
7	-1.5632e-...	-2.8422e-...	-1.4211e-...	-4.2633e-...	-1.4211e-...	0	-9.9476e-...	-1.4211e-...	
8	-4.2633e-...	8.5265e-14	1.1369e-13	8.5265e-14	1.1369e-13	1.7053e-13	0	1.7053e-13	

(4) 结果分析:

得到的 dif 矩阵元素的数量级在 e^{-13} ~ e^{-14} 量级, 因此可以验证在变换域上实现将像素值减去 128。

2.2 请编程实现二维 DCT, 并于 matlab 自带的库函数 dct2 比较是否一致

(1) 主要思想:

按照二维 DCT 的定义, 先生成 DCT 变换矩阵的每一行, 然后根据定义对输入的图像块进行 DCT 变换。

(2) 核心代码:

编程实现的 DCT 函数

```
function MatC = DCT(MatP)
% get 2d dct transform
% MatP is a square matrix
sizeP = size(MatP);
if sizeP(1) ~= sizeP(2)
    disp("Error! MatP is not a square matrix!");
    return;
end
dim = length(MatP);
MatD = zeros(dim,dim);
for i = 1 : 1 : dim
    if i == 1
        MatD(i,:) = sqrt(1/2)*ones(1,dim);
    else
        MatD(i,:) = cos(pi/(2*dim)*(i-1:2*(i-1):(i-1)*(2*dim-1)));
    end
end
MatC = 2/dim*MatD*MatP*(MatD.');
```

库函数 dct2 代码:

```
function b=dct2(arg1,mrows,ncols)
%DCT2 2-D discrete cosine transform.
% B = DCT2(A) returns the discrete cosine transform of A.
% The matrix B is the same size as A and contains the
% discrete cosine transform coefficients.
```



```

%
% B = DCT2(A,[M N]) or B = DCT2(A,M,N) pads the matrix A with
% zeros to size M-by-N before transforming. If M or N is
% smaller than the corresponding dimension of A, DCT2 truncates
% A.
%
% This transform can be inverted using IDCT2.
%
% Class Support
% -----
% A can be numeric or logical. The returned matrix B is of
% class double.
%
% Example
% -----
%     RGB = imread('autumn.tif');
%     I = rgb2gray(RGB);
%     J = dct2(I);
%     imshow(log(abs(J)),[]), colormap(gca,jet), colorbar
%
% The commands below set values less than magnitude 10 in the
% DCT matrix to zero, then reconstruct the image using the
% inverse DCT function IDCT2.
%
%     J(abs(J)<10) = 0;
%     K = idct2(J);
%     figure, imshow(I)
%     figure, imshow(K,[0 255])
%
% See also FFT2, IDCT2, IFFT2.
%
% Copyright 1992-2016 The MathWorks, Inc.
%
% References:
%     1) A. K. Jain, "Fundamentals of Digital Image
%        Processing", pp. 150-153.
%     2) Wallace, "The JPEG Still Picture Compression Standard",
%        Communications of the ACM, April 1991.

[m, n] = size(arg1);
% Basic algorithm.
if (nargin == 1),
    if (m > 1) && (n > 1),

```

```

        b = dct(dct(arg1).').';
        return;
    else
        mrows = m;
        ncols = n;
    end
end

% Padding for vector input.
a = arg1;
if nargin==2, ncols = mrows(2); mrows = mrows(1); end
mpad = mrows; npad = ncols;
if m == 1 && mpad > m, a(2, 1) = 0; m = 2; end
if n == 1 && npad > n, a(1, 2) = 0; n = 2; end
if m == 1, mpad = npad; npad = 1; end % For row vector.

% Transform.

b = dct(a, mpad);
if m > 1 && n > 1, b = dct(b.', npad).'; end

```

(3) 核心代码说明:

在自行编写的代码中, 输入矩阵为 MatP, 变换矩阵为 MatD, 输出处理后的矩阵为 MatC。在自行编写的程序中要求输入的矩阵必须是一个方阵, 否则将会抛出错误。

(4) 结果分析:

对比自己写的 DCT 函数与库函数 dct2 代码实现的方式并不一致。库函数可以实现一般矩阵的二维 dct 变化, 且可以对于输入自变量个数的不同进行函数的重载。在具体实现上, 自己编写的 DCT 函数是通过 for 循环控制生成二维 DCT 变化矩阵。而自带库函数 dct2 是通过一维 DCT 变换实现二维 DCT 变换。

2.3 若将 DCT 系数矩阵的右边四列系数全部置零，逆变换后的图像会发生什么变化？选取一块图验证，如果左边四列置零又会发生什么变化？

(1) 主要思想：

在进行 DCT 变换后，将得到的 8×8 矩阵 P 右边四列置零得到矩阵 P'，将 P' 进行 IDCT 变换即可得到所需要的右侧置零矩阵。同理对 P' 矩阵的左边四列置零进行相关的变换即可获得左侧置零处理后的矩阵。

(2) 核心代码：

图像分块函数 GraphicsDivide:

```
function [proc_graph,blocks,block_num,height,width] =  
GraphicsDivide(prim_graph)  
    prim_graph = double(prim_graph);  
    prim_size = size(prim_graph);  
    height = ceil(prim_size(1)/8);  
    width = ceil(prim_size(2)/8);  
    block_num = height*width;  
    proc_graph = zeros(height*8,width*8);  
    proc_graph(1:prim_size(1),1:prim_size(2)) = prim_graph;  
    blocks = zeros(8,8,block_num);  
    for i = 1 : 1 : 8*height  
        if i <= prim_size(1)  
            proc_graph(i,prim_size(2)+1:8*width) =  
prim_graph(i,prim_size(2))*ones(1,8*width-prim_size(2));  
        else  
            proc_graph(i,:) = proc_graph(prim_size(1),:);  
        end  
    end  
    for i = 1 : block_num  
        blocks(:, :, i) =  
proc_graph(8*floor((i-1)/width)+1:8*floor((i-1)/width)+8,8*mod((i-1),  
width)+1:8*mod((i-1),width)+8);  
    end  
end
```

图像恢复函数 BlockReform

```
function [image] = BlockReform(blocks,height,width)  
    block_size = size(blocks);  
    image = zeros(height*8,width*8);
```

```

        for i = 1 : 1 : block_size(3)

image(8*floor((i-1)/width)+1:8*floor((i-1)/width)+8,8*mod((i-1),width)+1:8*mod((i-1),width)+8) = blocks(:,:,i);

        end
    end
end

```

实现脚本:

```

load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat
hall_gray = double(hall_gray);
[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(hall_gray-128);
l_blocks = zeros(8,8,block_num);
r_blocks = zeros(8,8,block_num);
for i = 1 : 1 : block_num
    l_blocks(:,:,i) = DCT(blocks(:,:,i)).*[zeros(8,4),ones(8,4)];
    l_blocks(:,:,i) = IDCT(l_blocks(:,:,i));
    r_blocks(:,:,i) = DCT(blocks(:,:,i)).*[ones(8,4),zeros(8,4)];
    r_blocks(:,:,i) = IDCT(r_blocks(:,:,i));
end
image_rz = BlockReform(r_blocks,height,width);
image_rz = uint8(image_rz+128);
image_lz = BlockReform(l_blocks,height,width);
image_lz = uint8(image_lz+128);
figure;
imshow(image_rz);
figure;
imshow(image_lz);
imwrite(image_rz,'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapt
er_2\\RightZeroDCT.jpg');
imwrite(image_lz,'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapt
er_2\\LeftZeroDCT.jpg');

```

(3) 核心代码说明:

在 GraphicsDivide 函数中, 输入为原始灰度图像, 输出为分割后增补的图像 proc_graph, 分割后形成的图像块 blocks, 图像块数量 block_num, 处理后的图像高度 height, 处理后的图像高度 width。高度和宽度用该维度上的块数量来刻画。在此函数先将原始图像根据边缘的像素值拓展为最近的处理后图像, 处理后图像的高和宽方向的像素个数均能被

8 整除。然后按照每一行从左到右，从上到下扫描处理后的图像的每一个 8×8 块，将相应的块放到 blocks 数组中。Blocks 数组的前两个维度均为 8，表示该图像块的 8×8 个像素。第三个维度表示该图像块所对应的块序号。

在 BlockReform 函数中通过图像块数组按照图像块的扫描顺序即可恢复出处理前的图像。

在实现脚本中，DCT 系数右侧置零的图像为 image_rz，DCT 系数左侧置零的图像为 image_lz。

(4) 运行结果：

DCT 系数矩阵右侧置零后恢复出的图像为：



DCT 系数矩阵左侧置零后恢复出的图像为：



(5) 结果分析：

由运行结果可知，DCT 系数右侧置零后恢复出的图像和原图像之间几乎没有区别。这是因为图像经过 DCT 变化后，各个频率分量存放在了 DCT 系数矩阵中。原始图像中低频分量很多而高频分量很少，因此对于 DCT 系数右侧置零相当于过滤掉了图片中的高频分量，对图像几乎没有影响。DCT 系数左侧置零后恢复出的图像和原图像完全不同，这是因为将 DCT 系数矩阵的左侧元素置零后直接丢失掉了图像最主要的低频部分，因此不能恢复出原图像。

2.4 对 DCT 系数分别做转置、旋转 90° 、旋转 180° 操作(rot90)，逆变换后恢复的图像有什么变化，选取一块图验证

(1) 主要思想：

对经过 DCT 变换后的系数矩阵进行相应的转置、旋转 90°、旋转 180°等操作，对比处理器前和处理后图像左上角 8×8 块的图像矩阵。

(2) 核心代码:

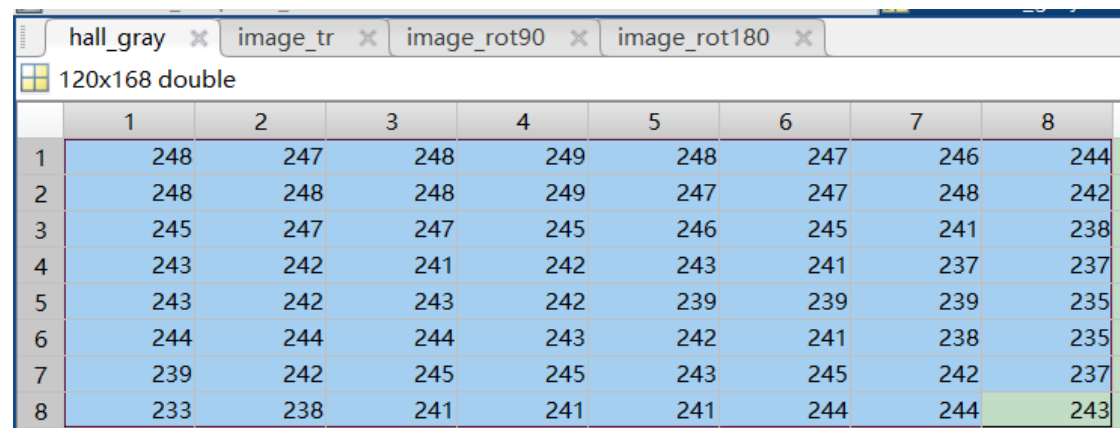
```
clear all, close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\hall.mat
hall_gray = double(hall_gray);
[proc_graph, blocks, block_num, height, width] =
GraphicsDivide(hall_gray-128);
tr_blocks = zeros(8,8,block_num);
rot90_blocks = zeros(8,8,block_num);
rot180_blocks = zeros(8,8,block_num);
temp = zeros(8,8);
for i = 1 : 1 : block_num
    temp = DCT(blocks(:,:,i));
    tr_blocks(:,:,i) = temp.';
    tr_blocks(:,:,i) = IDCT(tr_blocks(:,:,i));
    rot90_blocks(:,:,i) = rot90(temp,1);
    rot90_blocks(:,:,i) = IDCT(rot90_blocks(:,:,i));
    rot180_blocks(:,:,i) = rot90(temp,2);
    rot180_blocks(:,:,i) = IDCT(rot180_blocks(:,:,i));
end
image_tr = BlockReform(tr_blocks,height,width);
image_tr = uint8(image_tr+128);
figure;
imshow(image_tr);
image_rot90 = BlockReform(rot90_blocks,height,width);
image_rot90 = uint8(image_rot90+128);
figure;
imshow(image_rot90);
image_rot180 = BlockReform(rot180_blocks,height,width);
image_rot180 = uint8(image_rot180+128);
figure;
imshow(image_rot180);
imwrite(image_tr, 'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\TransferDCT.jpg');
imwrite(image_rot90, 'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\Rotate90DCT.jpg');
imwrite(image_rot180, 'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\Rotate180DCT.jpg');
```

(3) 核心代码说明:

上述代码的转置、旋转 90°以及旋转 180°操作均是针对经过 DCT 变换后的每一个 8×8 的块。

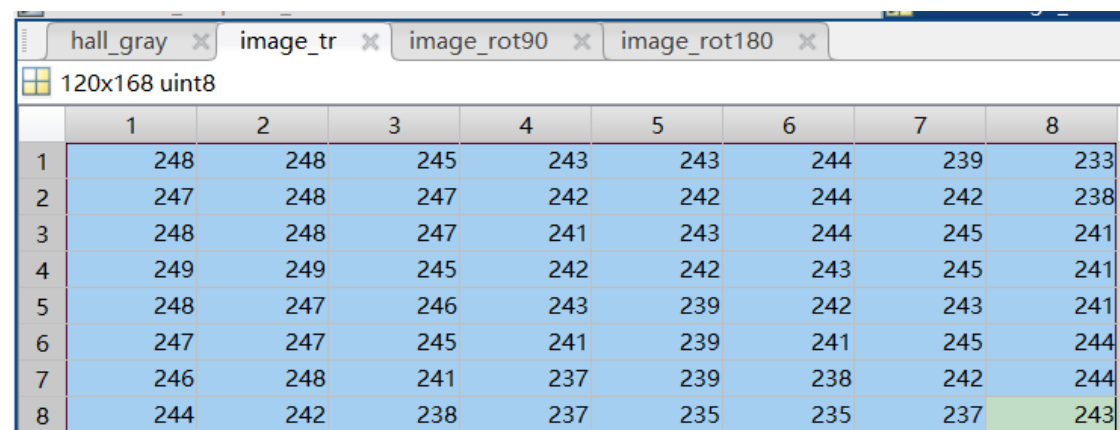
(4) 运行结果:

原始图像左上角 8×8 图像块矩阵:



	1	2	3	4	5	6	7	8
1	248	247	248	249	248	247	246	244
2	248	248	248	249	247	247	248	242
3	245	247	247	245	246	245	241	238
4	243	242	241	242	243	241	237	237
5	243	242	243	242	239	239	239	235
6	244	244	244	243	242	241	238	235
7	239	242	245	245	243	245	242	237
8	233	238	241	241	241	244	244	243

DCT 转置后图像左上角 8×8 图像块矩阵:



	1	2	3	4	5	6	7	8
1	248	248	245	243	243	244	239	233
2	247	248	247	242	242	244	242	238
3	248	248	247	241	243	244	245	241
4	249	249	245	242	242	243	245	241
5	248	247	246	243	239	242	243	241
6	247	247	245	241	239	241	245	244
7	246	248	241	237	239	238	242	244
8	244	242	238	237	235	235	237	243

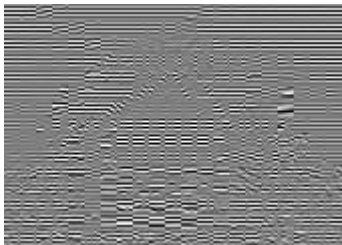
DCT 转置后图像:



DCT 旋转 90°后图像左上角 8×8 图像块矩阵:

hall_gray × image_tr × image_rot90 × image_rot180 ×								
120x168 uint8								
	1	2	3	4	5	6	7	8
1	162	161	159	160	160	160	157	154
2	34	34	36	37	38	37	41	46
3	255	255	255	255	255	255	255	255
4	0	0	0	0	0	0	0	0
5	255	255	255	255	255	255	255	255
6	0	0	0	0	0	0	0	0
7	219	219	213	212	213	211	213	219
8	97	100	99	98	101	100	100	96

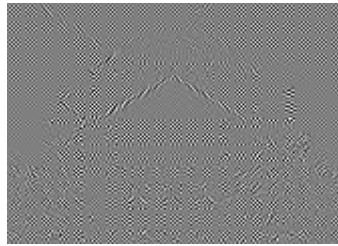
DCT 旋转 90°后图像:



DCT 旋转 180°后图像左上角 8×8 图像块矩阵:

120x168 uint8								
	1	2	3	4	5	6	7	8
1	137	102	167	82	175	90	153	118
2	101	203	17	255	0	240	54	152
3	166	18	255	0	255	0	232	94
4	84	254	0	255	0	255	10	169
5	172	4	255	0	255	0	246	89
6	92	232	0	255	0	255	29	161
7	148	65	231	3	253	18	199	104
8	122	149	95	168	85	166	100	139

DCT 旋转 180°后图像:



(5) 结果分析:

由运行结果可知, 转置后的 DCT 系数矩阵经过 IDCT 变换后得到的新的图像块矩阵是原来的转置。这是由于 DCT 变换和 IDCT 变换中的变换矩阵是正交阵, 因此经过逆变换后的矩阵是原始矩阵的转置, 运行结果验证了这一点, 从整块图像也可以看出有大礼堂的样子只是图像块的拼接不同。而将 DCT 系数矩阵经过旋转变换后经过 IDCT 变换得到的图像不能恢复出原始图像的信息。

2.5 若认为差分编码是一个离散时间系统, 绘制出这个系统的频率响应, 说明其滤波器类型。DC 系数先进行差分编码再进行熵编码, 说明 DC 系数的哪种频率分量更多?

(1) 主要思想:

通过差分编码的规则, 将其抽象为一个离散时间系统。通过 freqz 函数绘制系统的频率响应。

(2) 核心代码:

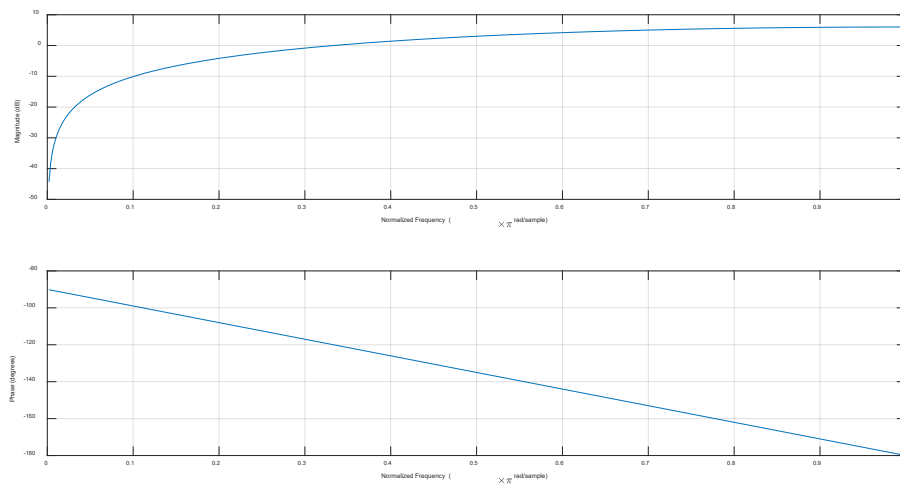
```
clear all, close all, clc;  
a = [1];  
b = [-1, 1];  
freqz(b, a);
```

(3) 核心代码说明:

由差分编码规则可以将该系统抽象为 $y(n) = x(n-1) - x(n)$, 因此滤波器的分子系数为 [1], 分母系数为 [-1, 1]。

(4) 运行结果：

该系统的频率响应为(上为幅频响应，下为相频响应)



(5) 结果分析：

由幅频响应曲线可知，该系统是一个高通滤波器。因此说明 DC 系数中高频分量更多。

2.6 DC 预测误差取值和 Category 有什么关系？如何利用预测误差计算处 Category？

(1) 主要思想：

由题目中表 2.2 可得,DC 预测误差的绝对值越大则对应的 Category 的值越大，且每一个 Category 所对应的 DC 误差范围大小同 Category 的大小成指数关系。在数值上，DC 预测误差与种类 Category 的关系为

$$Category = \text{ceil}(\log_2(\text{abs}(DC_error) + 1))$$

2.7 可以用多种算法实现 ZigZag 扫描，设计一种最佳的方法

(1) 主要思想：

在本题中提供了两种思路实现 8×8 矩阵的 ZigZag 扫描。其中第一种方式为直接映射，通过查表的方式输出 ZigZag 扫描后的 AC 系数流。第二种方法是通过 For 循环的方法提取出每个斜对角线中的元素，将提取出的向量按照扫描的方向进行拼接即可得到 ZigZag 扫描后的 AC 流。

(2) 核心代码:

查表法:

```
function [seq_AC] = ZigZag(mat_AC)
    seq_AC =
    [mat_AC(1,2),mat_AC(2,1),mat_AC(3,1),mat_AC(2,2),mat_AC(1,3),mat_AC(1,4),mat_AC(2,3),mat_AC(3,2),mat_AC(4,1),mat_AC(5,1),mat_AC(4,2),mat_AC(3,3),mat_AC(2,4),mat_AC(1,5),mat_AC(1,6),mat_AC(2,5),mat_AC(3,4),mat_AC(4,3),mat_AC(5,2),mat_AC(6,1),mat_AC(7,1),mat_AC(6,2),mat_AC(5,3),mat_AC(4,4),mat_AC(3,5),mat_AC(2,6),mat_AC(1,7),mat_AC(1,8),mat_AC(2,7),mat_AC(3,6),mat_AC(4,5),mat_AC(5,4),mat_AC(6,3),mat_AC(7,2),mat_AC(8,1),mat_AC(8,2),mat_AC(7,3),mat_AC(6,4),mat_AC(5,5),mat_AC(4,6),mat_AC(3,7),mat_AC(2,8),mat_AC(3,8),mat_AC(4,7),mat_AC(5,6),mat_AC(6,5),mat_AC(7,4),mat_AC(8,3),mat_AC(8,4),mat_AC(7,5),mat_AC(6,6),mat_AC(5,7),mat_AC(4,8),mat_AC(5,8),mat_AC(6,7),mat_AC(7,6),mat_AC(8,5),mat_AC(8,6),mat_AC(7,7),mat_AC(6,8),mat_AC(7,8),mat_AC(8,7),mat_AC(8,8)];
end
```

循环控制法:

```
function [seq_AC] = ZigZag_1(mat_AC)
    temp_AC = rot90(mat_AC);
    AC_size = size(temp_AC);
    seq_AC = [];
    for i = -AC_size(1)+2 : 1 : AC_size(2)-1
        temp = diag(temp_AC,i);
        if mod(abs(i),2) == 0
            seq_AC = [seq_AC;temp];
        else
            seq_AC = [seq_AC;flipud(temp)];
        end
    end
    seq_AC = seq_AC.';
end
```

(3) 核心代码说明:

ZigZag 查表实现中输出 AC 流的顺序是按照题目中 2.6 的顺序依次扫描得到。在速度上由于是直接映射因此速度更快。For 循环控制法是先将原始 AC 矩阵旋转 90° ，然后依次取旋转后的矩阵的各阶对角线元素，按照响应的扫描顺序将这些元素拼接在一起。

2.8 对测试图像进行分块，DCT 和量化，将量化后的系数写成矩阵的形式，其中每一列为每个块经过 ZigZag 扫描后的列矢量，第一行为各个块的 DC 系数。

(1) 主要思想：

先通过 GraphicsDivide 函数将原始图像进行分块，并进行每个块的 DCT 变化。将处理后的每一个块点除量化矩阵得到量化后的矩阵。按照题目中要求的顺序将量化后的矩阵 reshape 为所需的矩阵格式。

(2) 核心代码：

```
clear all, close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\JpegCoeff.mat
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\hall.mat
hall_gray = double(hall_gray);
[proc_graph, blocks, block_num, height, width] =
GraphicsDivide(hall_gray-128);
mat_DCT = zeros(8,8,block_num);
quanti_mat = zeros(64,block_num);
for i = 1 : 1 : block_num
    mat_DCT(:,:,i) = DCT(blocks(:,:,i));
    quanti_mat(:,i) = [mat_DCT(1,1,i); (ZigZag(mat_DCT(:,:,i))).'];
end
quanti_mat = int16(quanti_mat);
```

(3) 核心代码说明：

从文件中读取 hall_gray 图像后需要将图像的 uint8 类型转化为 double 类型进行计算。最后得到的 DCT 量化系数矩阵需要保留为 int16 类型。其余函数的具体操作如报告之前的内容所示。

(4) 结果分析：

最终得到了一个大小为 64×315 的量化矩阵 quanti_mat。其中量化矩阵的第一行为列所对应的块(列序号等于块序号)的 DC 系数。每一列的 2~64 行对应于该列所在块经过 ZigZag 扫描后的 AC 流。

2.9 请实现本章中的 JPEG 编码，输出为 DC 系数的码流、AC 系数的码流、图像高度与图像宽度，将这四个变量写进 jpegcodes.mat 文件中

(1) 主要思想:

在 2.8 的基础上对将图像的 DC 量化码流与 AC 量化码流提取出来, 分别按照 DC 预测误差 Huffman 编码规则以及 AC 系数块的 Huffman 编码规则进行编码, 即可得到相应的 DC 流以及 AC 流编码, 将编码同图像宽度以及图像高度写入 jpegcodes.mat 文件中。

(2) 核心代码:

DC 编码函数 EncodeDC:

```
function [code_DC] = EncodeDC(seq_DC_error)
    len_DC = length(seq_DC_error);
    code_DC = [];
    for i = 1 : 1 : len_DC
        exp_error = seq_DC_error(i);
        category = ceil(log2(abs(exp_error)+1));
        if category == 0
            huffman = '00';
        elseif category == 1
            huffman = '010';
        elseif category == 2
            huffman = '011';
        elseif category == 3
            huffman = '100';
        elseif category == 4
            huffman = '101';
        elseif category == 5
            huffman = '110';
        elseif category == 6
            huffman = '1110';
        elseif category == 7
            huffman = '11110';
        elseif category == 8
            huffman = '111110';
        elseif category == 9
            huffman = '1111110';
        elseif category == 10
            huffman = '11111110';
        elseif category == 11
            huffman = '111111110';
        end
        code_DC = [code_DC, huffman];
    end
```

```

        code_DC = [code_DC,huffman,Dec2Complement(exp_error)];
    end
end

```

AC 编码函数 EncodeAC

```

function [code_AC] = EncodeAC(seq_AC)
    code_AC = []; % 1:non_zeros sequence,0:zeros sequence
    run = 0;
    last_non_zero = 63;
    for i = 63 : -1 : 1
        if seq_AC(i) ~= 0
            last_non_zero = i;
            break;
        end
        if (i == 1)&&(seq_AC(i) == 0)
            last_non_zero = -1;
            break;
        end
    end
    for i = 1 : 1 : last_non_zero
        if seq_AC(i) ~= 0
            code_AC =
[code_AC,RuleAC(run,seq_AC(i)),Dec2Complement(seq_AC(i))];
            run = 0;
        else
            run = run + 1;
            if (run == 16)&&(i ~= 63)
                code_AC = [code_AC,'11111111001'];
                run = 0;
            end
        end
    end
    if last_non_zero ~= 63
        code_AC = [code_AC,'1010'];
    end
end

```

AC 编码规则函数 RuleAC

```

function [single_code_AC] = RuleAC(run,seq_val)

```

```

load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\JpegCoeff.mat
if seq_val == 0
    size = 0;
else
    size = floor(log2(abs(seq_val)))+1;
end
code_end = 3+ACTAB(run*10+size,3);
single_code_AC = char((ACTAB(run*10+size,4:code_end))+48);
end

```

辅助函数：十进制转一补码函数 Dec2Complement

```

function [complement] = Dec2Complement(dec_in)

if dec_in == 0
    N = 1;
else
    N = floor(log2(abs(dec_in)))+1;
end
if dec_in >= 0
    complement = dec2bin(dec_in,N);
else
    complement = dec2bin(2^N-1+dec_in,N);
end
end

```

编码脚本：

```

clear all, close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\JpegCoeff.mat
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat
hall_gray = double(hall_gray);
[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(hall_gray-128);
quanti_mat_DCT = zeros(8,8,block_num);
quanti_mat = zeros(64,block_num);
for i = 1 : 1 : block_num
    quanti_mat_DCT(:,:,i) = round(DCT(blocks(:,:,i))./QTAB);
end

```

```

    quanti_mat(:,i) =
[quanti_mat_DCT(1,1,i);(ZigZag_1(quanti_mat_DCT(:, :, i))).'];
end
quanti_mat = int16(quanti_mat);
seq_DC = zeros(1,block_num);
seq_AC = zeros(block_num,63);
code_AC = [];
for i = 1 : 1 : block_num
    if i == 1
        seq_DC(i) = quanti_mat(1,i);
    else
        seq_DC(i) = quanti_mat(1,i-1)-quanti_mat(1,i);
    end
    seq_AC(i,:) = (quanti_mat(2:64,i)).';
end
code_DC = EncodeDC(seq_DC);
for i = 1 : 1 : block_num
    code_AC = [code_AC,EncodeAC(seq_AC(i,:))];
end
code_AC = logical(str2num(code_AC(:))');
code_DC = logical(str2num(code_DC(:))');
save('C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\jpegcodes.mat','code_DC','code_AC','height','width');

```

(3) 核心代码说明:

进行 DC 编码时，直接将整个图像的所有 DC 系数经过差分运算后送入到 EncodeDC 函数进行编码。而进行 AC 系数编码时，是将每一个块对应的 AC 流分别送入 EncodeAC 函数进行编码。将每个块得到的 AC 编码结果按照块的顺序首尾相接即可获得 AC 编码。需要说明的是，存放到 jpegcodes 文件中的宽度和高度是图像经过处理后矩阵的列和行对应的块数。

2.10 计算图像压缩比(输入文件长度/输出码流长度)，注意转化为相同进制

(1) 主要思想:

运行 2.9 中脚本后，得到 code_DC 为长度为 1×2054 的 logical 数据，code_AC 为长度为 1×23073 的 logical 数据类型。原始图像的像素点数为 120×168，因此输入文件长度为

$120 \times 168 \times 8 = 161280 \text{bits}$ 。输出码流的长度为 $1 \times 2054 + 1 \times 23073 = 25127 \text{bits}$ 。因此文

图像的压缩比为 $\frac{161280}{25127} = 6.42$

2.11 请实现本章中的 JPEG 解码,输入是生成的 jpegcodes.mat 文件。分别用客观(PSNR)和主观方式评价编译码效果

(1) 主要思想:

实现 JPEG 解码过程中最关键的为将编码后的 DC 码流和 AC 码流无失真解码恢复。在本题中利用 Huffman 编码的特点通过查找 huffman 树的根节点对 AC 以及 DC 码流进行解码。将解码得到的 DC 与 AC 码流经过相应的反 ZigZag, 拼接, IDCT 以及反量化, 图像块复原等操作即可获得解码后的图像。

(2) 核心代码:

DC 解码函数 DecodeDC:

```
function [dct_DC_seq] = DecodeDC(seq_DC)
    load
    C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\HuffmanTreeD
    C.mat;
    dct_DC_seq = [];
    i = 1;
    cursor = 1;
    while i <= length(seq_DC)
        if seq_DC(i) == 0
            cursor = 2*cursor;
            if HuffTree(cursor,3) == 1
                if HuffTree(cursor,4) == 0
                    dct_DC_seq = [dct_DC_seq,0];
                    i = i + 2;
                    cursor = 1;
                    continue;
                else
                    len = HuffTree(cursor,4);
                    dct_DC_seq =
[dct_DC_seq,Complement2Dec(seq_DC(i+1:i+len))];
                    i = i + len + 1;
                    cursor = 1;
                    continue;
                end
            end
        end
    end
```

```

        else
            i = i + 1;
            continue;
        end
    else
        cursor = 2*cursor+1;
        if HuffTree(cursor,3) == 1
            if HuffTree(cursor,4) == 0
                dct_DC_seq = [dct_DC_seq,0];
                i = i + 2;
                cursor = 1;
                continue;
            else
                len = HuffTree(cursor,4);
                dct_DC_seq =
[dct_DC_seq,Complement2Dec(seq_DC(i+1:i+len))];
                i = i + len + 1;
                cursor = 1;
                continue;
            end
        else
            i = i + 1;
            continue;
        end
    end
end
end
for i = 2 : 1 : length(dct_DC_seq)
    dct_DC_seq(i) = dct_DC_seq(i-1)-dct_DC_seq(i);
end
end
end

```

AC 解码函数 DecodeAC:

```

function [dct_AC_seq] = DecodeAC(seq_AC)
    load
C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\HuffmanTreeA
C.mat;
    dct_AC_seq = [];
    i = 1;
    cursor = 1;
    cnt = 0;
    while i<= length(seq_AC)

```

```

if seq_AC(i) == 0
    cursor = 2*cursor;
    if HuffTree(cursor,3) == 1
        size = HuffTree(cursor,5);
        run = HuffTree(cursor,4);
        if size > 0
            amplitude = Complement2Dec(seq_AC(i+1:i+size));
            dct_AC_seq = [dct_AC_seq,zeros(1,run),amplitude];
            i = i + 1 + size;
            cnt = cnt + run + 1;
            cursor = 1;
        elseif (size == 0)&&(run == 15)
            dct_AC_seq = [dct_AC_seq,zeros(1,16)];
            i = i + 1;
            cnt = cnt + 16;
            cursor = 1;
        else
            dct_AC_seq = [dct_AC_seq,zeros(1,63-cnt)];
            i = i + 1;
            cnt = 0;
            cursor = 1;
            continue;
        end
        if cnt == 63
            cnt = 0;
        end
    else
        i = i + 1;
        continue;
    end
else
    cursor = 2*cursor + 1;
    if HuffTree(cursor,3) == 1
        size = HuffTree(cursor,5);
        run = HuffTree(cursor,4);
        if size > 0
            amplitude = Complement2Dec(seq_AC(i+1:i+size));
            dct_AC_seq = [dct_AC_seq,zeros(1,run),amplitude];
            i = i + 1 + size;
            cnt = cnt + run + 1;
            cursor = 1;
        elseif (size == 0)&&(run == 15)
            dct_AC_seq = [dct_AC_seq,zeros(1,16)];

```

```

        i = i + 1;
        cnt = cnt + 16;
        cursor = 1;
    else
        dct_AC_seq = [dct_AC_seq,zeros(1,63-cnt)];
        i = i + 1;
        cnt = 0;
        cursor = 1;
        continue;
    end
    if cnt == 63
        cnt = 0;
    end
    else
        i = i + 1;
        continue;
    end
end
end
end
end

```

生成 DC 解码 Huffman 树函数 HuffTreeBuildDC:

```

function [] = HuffTreeBuildDC()
    load C:\\Users\\HJSF\\Desktop\\Dynamic\\图像处理大作业\\图像处理所需资源\\JpegCoeff.mat;
    HuffTree = zeros(2^10-1,4);
    %dim 1 : left child;
    %dim 2 : right child;
    %dim 3 : leaf flag; 1 for leaf
    %dim 4 : category

    for i = 1 : 1 : 2^10 - 1
        if i < 2^9
            HuffTree(i,1) = 2*i;
            HuffTree(i,2) = 2*i+1;
        else
            HuffTree(i,1) = -1;
            HuffTree(i,2) = -1;
        end
    end
    root_num = 1;
    for i = 1 : 1 : 12
        code_len = DCTAB(i,1);
    end
end

```

```

DC_code = DCTAB(i,2:1+code_len);
current_num = root_num;
for j = 1 : 1 : code_len
    if DC_code(j) == 0
        current_num = 2*current_num;
    else
        current_num = 2*current_num + 1;
    end
    if j == code_len
        HuffTree(current_num,3) = 1;
        HuffTree(current_num,4) = i-1;
    end
end
end

save('C:\\Users\\HJSF\\Desktop\\Dynamic\\PP_chapter_2\\HuffmanTreeDC.
mat','HuffTree');
end

```

生成 AC 解码 Huffman 树函数 HuffTreeBuildAC:

```

function [] = HuffTreeBuildAC()
    load C:\\Users\\HJSF\\Desktop\\Dynamic\\图像处理大作业\\图像处理所需资源\\JpegCoeff.mat;
    HuffTree = zeros(2^17-1,5);
    %dim 1 : left child;
    %dim 2 : right child;
    %dim 3 : leaf flag; 1 for leaf
    %dim 4 : run
    %dim 5 : size

    for i = 1 : 1 : 2^17 - 1
        if i < 2^16
            HuffTree(i,1) = 2*i;
            HuffTree(i,2) = 2*i+1;
        else
            HuffTree(i,1) = -1;
            HuffTree(i,2) = -1;
        end
    end

    root_num = 1;
    for i = 1 : 1 : length(ACTAB)
        code_len = ACTAB(i,3);
        AC_code = ACTAB(i,4:3+code_len);
    end
end

```

```

        current_num = root_num;
        for j = 1 : 1 : code_len
            if AC_code(j) == 0
                current_num = 2*current_num;
            else
                current_num = 2*current_num + 1;
            end
            if j == code_len
                HuffTree(current_num,3) = 1;
                HuffTree(current_num,4) = ACTAB(i,1);
                HuffTree(current_num,5) = ACTAB(i,2);
            end
        end
    end
    HuffTree(26,3:5) = [1,0,0];
    HuffTree(4089,3:5) = [1,15,0];

    save('C:\\Users\\HJSF\\Desktop\\Dynamic\\PP_chapter_2\\HuffmanTreeAC.
    mat','HuffTree');
end

```

辅助函数 Complement2Dec:

```

function [dec_out] = Complement2Dec(complement)
    len = length(complement);
    if complement(1) == 0
        for i = 1 : 1 : len
            complement(i) = 1 - complement(i);
        end
        complement = char(complement+48);
        dec_out = -bin2dec(complement);
    else
        complement = char(complement+48);
        dec_out = bin2dec(complement);
    end
end

```

反 ZigZag 变换:

```

function [dct_AC_mat] = InverseZiZag(dct_AC_seq)
    dct_AC_mat = zeros(8,8);

```

```

    dct_AC_mat =
    [0,dct_AC_seq(1),dct_AC_seq(5),dct_AC_seq(6),dct_AC_seq(14),dct_AC_se
    q(15),dct_AC_seq(27),dct_AC_seq(28);dct_AC_seq(2),dct_AC_seq(4),dct_A
    C_seq(7),dct_AC_seq(13),dct_AC_seq(16),dct_AC_seq(26),dct_AC_seq(29),
    dct_AC_seq(42);dct_AC_seq(3),dct_AC_seq(8),dct_AC_seq(12),dct_AC_seq(
    17),dct_AC_seq(25),dct_AC_seq(30),dct_AC_seq(41),dct_AC_seq(43);dct_A
    C_seq(9),dct_AC_seq(11),dct_AC_seq(18),dct_AC_seq(24),dct_AC_seq(31),
    dct_AC_seq(40),dct_AC_seq(44),dct_AC_seq(53);dct_AC_seq(10),dct_AC_se
    q(19),dct_AC_seq(23),dct_AC_seq(32),dct_AC_seq(39),dct_AC_seq(45),dct
    _AC_seq(52),dct_AC_seq(54);dct_AC_seq(20),dct_AC_seq(22),dct_AC_seq(3
    3),dct_AC_seq(38),dct_AC_seq(46),dct_AC_seq(51),dct_AC_seq(55),dct_AC
    _seq(60);dct_AC_seq(21),dct_AC_seq(34),dct_AC_seq(37),dct_AC_seq(47),
    dct_AC_seq(50),dct_AC_seq(56),dct_AC_seq(59),dct_AC_seq(61);dct_AC_se
    q(35),dct_AC_seq(36),dct_AC_seq(48),dct_AC_seq(49),dct_AC_seq(57),dct
    _AC_seq(58),dct_AC_seq(62),dct_AC_seq(63)];
end

```

将DCT矩阵还原函数DctMatReform:

```

function [dct_mat] = DctMatReform(dct_AC_mat,dct_DC_seq)
    mat_num = length(dct_DC_seq);
    dct_mat = zeros(8,8,mat_num);
    dct_mat = dct_AC_mat;
    for i = 1 : 1 : mat_num
        dct_mat(1,1,i) = dct_DC_seq(i);
    end
end

```

图像复原函数ImageReform:

```

function [image] = ImageReform(dct_mat,height,width)
    load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像
    处理所需资源\\JpegCoeff.mat;
    mat_size = size(dct_mat);
    mat_num = mat_size(3);
    image_blocks = zeros(8,8,mat_num);
    for i = 1 : 1 : mat_num
        image_blocks(:, :, i) = IDCT(dct_mat(:, :, i).*QTAB);
    end
    for i = 1 : 1 : mat_num
        image(floor((i-1)/width)*8+1:floor((i-1)/width)*8+8,mod(i-1,width)*8+
        1:mod(i-1,width)*8+8) = image_blocks(:, :, i);
    end
end

```

```

end
image = image + 128;
end

```

JPEG解码实现脚本:

```

clear all, close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat;
load
C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\jpegcodes.ma
t;
[dct_DC_seq] = DecodeDC(code_DC);
[dct_AC_seq] = DecodeAC(code_AC);
block_num = height*width;
dct_AC_mat = zeros(8,8,block_num);
for i = 1 : 1 : block_num
    dct_AC_mat(:,:,i) = InverseZiZag(dct_AC_seq(63*(i-1)+1:63*i));
end
[dct_mat] = DctMatReform(dct_AC_mat,dct_DC_seq);
[image] = ImageReform(dct_mat,height,width);
imwrite(uint8(image),'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_c
hapter_2\\my.jpg');
imshow(uint8(image));
prim_size = size(hall_gray);
pixel_num = prim_size(1)*prim_size(2);
MSE =
1/pixel_num*sum(sum((image-double(hall_gray)).*(image-double(hall_gra
y)))));
PSNR = 10*log10(255^2/MSE);
display(PSNR);

```

(3) 核心代码说明:

在进行解码前应先执行 HuffTreeBuildDC 以及 HuffTreeBuildAC 函数。在建立 DC 以及 AC 编码 Huffman 树时, 采用的是通过数组的方式建立二叉树, 通过坐标的方式来访问左孩子以及右孩子。当对应节点代表的码为码集中的码时, 则置标志位。在解码访问 Huffman 树的过程中, 同样采用坐标的方式对相应的节点进行访问, 若当前访问节点为码集中的一个码时, 则得到对应的 DC 以及 AC 元素或序列。此时, 停止访问, cursor 回到查找树的根节点, 开始下一次访问。

(4) 运行结果：

得到 JPEG 解码后的图像为：



计算得到的 PSNR 为：

```
命令行窗口
PSNR =
    31.1771
```

(5) 结果分析：

通过直观判断，解码得到的图像和原始图像几乎一样，肉眼分辨不出。通过客观 PSNR 方法判断得到 $PSNR = 31.1771 > 1$ ，也可以说明原始图像和解码后的图像差别很小。

2.12 将量化步长减小为原来的一般，重新编解码。同标准量化步长的情况比较压缩比与图像质量。

(1) 主要思想：

将量化矩阵中的每一个元素变为原来的一半，其余操作与函数同 2.11。

(2) 运行结果：

输出的图像为：



计算得到的 PSNR 如下：

```
命令行窗口

PSNR =

    34.2084

fx >>
```

得到的 DC_code 长度为 2421

得到的 AC_code 长度为 34156

(3) 结果分析：

压缩比为 4.409。因为量化步长减小，因此得到的量化后的 DCT 矩阵其中的元素绝对值越大，需要更多的比特来编码，从而导致压缩比下降。同时由于量化步长减小为了原来的一半，量化的精度有所上升，因此经过 JPEG 编解码后的图像质量有所上升。但是对于用户而言，图像质量上升的程度并不明显，然而压缩比下降的更多。因此折衷考虑应该仍使用原来的量化步长。

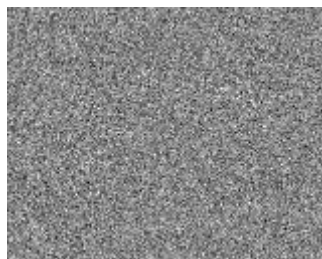
2.13 在看电视时经常能够雪花图像，对其进行编解码，同测试图像对比压缩比与图像质量，分析解释比较结果。

(1) 主要思想：

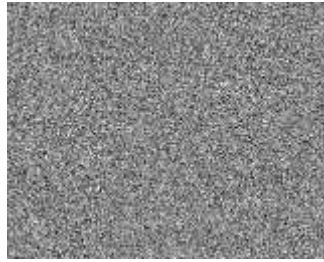
将 hall_gray 图像换成 snow 图像，其余操作同 2.11 中函数。

(2) 运行结果：

原始的雪花图像：



处理后的雪花图像：



计算得到的 PSNR:

```
命令行窗口
PSNR =
    22.9249
fx >>
```

得到的 code_DC 长度：1456

得到的 code_AC 长度：43367

(3) 结果分析：

压缩比为 3.655，远小于普通图像的压缩比 6.42,这是因为普通图像中低频分量较多，而雪花图像中高频分量较多，导致 AC 量化过程中很难出现连着的 0，因此 AC 码流较长。计算得到的 PSNR 为 22.92，客观效果并不好，这是因为雪花图像中高频分量较多导致在量化过程中有着较大的量化误差。

信息隐藏：

3.1 实现本章介绍的空域隐藏方法和提取方法。验证其抗 JPEG 编码的能力。

(1) 主要思想：

待隐藏信息为 signal and system is very fun。先将该信息隐藏图像的左上角，提取信息直接通过加密图像左上角区域进行提取即可。之后对加密图像进行正常的编解码操作。解码后，读取解码后的图像相应位置处的信息，验证其抗 JPEG 编码的能力。

(2) 核心代码:

空域隐藏与提取方法脚本实现:

```
clear all,close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat;
[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(hall_gray);
origin_info = 'signal and system is very fun';
info_bin = dec2bin(double(origin_info));
info_size = size(info_bin);
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        temp = dec2bin(proc_graph(i,j));
        temp(length(temp)) = info_bin(i,j);
        proc_graph(i,j) = bin2dec(temp);
    end
end
imwrite(uint8(proc_graph),'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\
\\PP_chapter_3\\figure3.1(加密).jpg');
info_trans_1 = InfoTranslate(proc_graph,info_size);
display(info_trans_1);

[code_AC,code_DC] = ImageEncode(proc_graph);
[image_hide_decode] = ImageDecode(code_AC,code_DC,height,width);
imwrite(uint8(image_hide_decode),'C:\\Users\\HJSF\\Desktop\\Dynamic\\
matlab\\PP_chapter_3\\figure3.1(解码).jpg');
pixel_num = 64*block_num;
MSE =
1/pixel_num*sum(sum((proc_graph-double(hall_gray)).*(proc_graph-doubl
e(hall_gray))));
PSNR = 10*log10(255^2/MSE);
display(PSNR);
[info_trans_2] = InfoTranslate(image_hide_decode,info_size);
display(info_trans_2);
```

ImageEncode 函数:

```
function [code_AC,code_DC] = ImageEncode(image)
    image = double(image);
    load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像
处理所需资源\\JpegCoeff.mat;
```

```

[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(image-128);
quanti_mat_DCT = zeros(8,8,block_num);
quanti_mat = zeros(64,block_num);
for i = 1 : 1 : block_num
    quanti_mat_DCT(:,:,i) = round(DCT(blocks(:,:,i))./QTAB);
    quanti_mat(:,i) =
[quanti_mat_DCT(1,1,i);(ZigZag(quanti_mat_DCT(:,:,i))).'];
end
quanti_mat = int16(quanti_mat);
seq_DC = zeros(1,block_num);
seq_AC = zeros(block_num,63);
code_AC = [];
for i = 1 : 1 : block_num
    if i == 1
        seq_DC(i) = quanti_mat(1,i);
    else
        seq_DC(i) = quanti_mat(1,i-1)-quanti_mat(1,i);
    end
    seq_AC(i,:) = (quanti_mat(2:64,i)).';
end
code_DC = EncodeDC(seq_DC);
for i = 1 : 1 : block_num
    code_AC = [code_AC,EncodeAC(seq_AC(i,:))];
end
end

```

ImageDecode 函数:

```

function [image_decode] = ImageDecode(code_AC,code_DC,height,width)
code_AC = str2num(code_AC(:))';
code_DC = str2num(code_DC(:))';
[dct_AC_seq] = DecodeAC(code_AC);
[dct_DC_seq] = DecodeDC(code_DC);
block_num = height*width;
dct_AC_mat = zeros(8,8,block_num);
for i = 1 : 1 : block_num
    dct_AC_mat(:,:,i) = InverseZiZag(dct_AC_seq(63*(i-1)+1:63*i));
end
[dct_mat] = DctMatReform(dct_AC_mat,dct_DC_seq);
[image_decode] = ImageReform(dct_mat,height,width);
image_decode = uint8(image_decode);
end

```

InfoTranslate 函数:

```
function [info_out] = InfoTranslate(image_hide_decode,info_size)
    info_out = [];
    for i = 1 : 1 : info_size(1)
        info_out =
[info_out,char(bin2dec(char(mod(image_hide_decode(i,1:info_size(2)),2
)+48)))];
    end
end
```

(3) 核心代码说明：

在实现空域隐藏时，直接将信息的各个 bit 位取代相应位置图像的最后一个 bit 位。ImageEncode 函数实现了将嵌密后的图像编码位 AC 码流与 DC 码流。ImageDecode 实现将编码码流转化为解码后的图像。读取解码后图像相应位置处的最后一个 bit 位即可还原出信息。

(4) 运行结果：

加密后的图像：



经过 JPEG 解码后的图像：



信息解密结果：

```
命令行窗口

info_trans_1 =

    'signal and system is very fun'

PSNR =

    70.7225

info_trans_2 =

    '1'
    9, b CF FC zsG#Ax K_5 '
```

(5) 结果分析：

由运行结果可知，加密后的图像和解密后的图像和原图像基本没有什么区别，达成了加密过程的隐蔽性要求，同时直接从加密后的图像上提取信息能够完全复原原始信息。但是通过解码图像得到的信息为乱码，和原始信息完全不同，因此空域加密抗 JPEG 编码能力很差，不适用于需要压缩的图像信息隐藏。

3.2 依次实现三种变换域隐藏方法和提取方法，分析嵌密方法的隐秘性以及嵌密后 JPEG 图像质量的变化和压缩比的变化。

(1) 主要思想：

三种加密过程均发生在 DCT 量化之后，熵编码之前。第一种加密方式为用信息为逐一替代所有量化后的 DCT 系数最低位。第二种加密方式为将信息位逐一替换位量化后的 DCT 系数最低位，在本实验中为替换掉 DC 系数位。第三种嵌密方式为用 ± 1 表示信息，将信息为添加在 ZigZag 顺序最后一个非零 DCT 系数之后，若最后一个系数不为零，则用信息为替换该位。在 DCT 域加密后通过 JPEG 编解码得到加密后的图像。再将加密后的图像经过 DCT 变换以及量化即可在相应位置提取隐藏信息。

(2) 核心代码：

加密方式 1 脚本：

```

clear all,close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\hall.mat;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理
所需资源\\JpegCoeff.mat;
hall_gray = double(hall_gray);
[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(hall_gray-128);
origin_info = 'signal and system is very fun';
info_bin = dec2bin(double(origin_info));
info_size = size(info_bin);
quanti_mat_DCT = zeros(8,8,block_num);
quanti_mat = zeros(64,block_num);
for i = 1 : 1 : block_num
    quanti_mat_DCT(:,:,i) = round(DCT(blocks(:,:,i))./QTAB);
end
quanti_mat = int16(quanti_mat);
quanti_mat_DCT = int16(quanti_mat_DCT);
list = [];
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        num = info_size(2)*(i-1)+j;
        index_3 = floor((num-1)/64)+1;
        index_1 = floor((num - 64*(index_3-1)-1)/8)+1;
        index_2 = mod(num - 64*(index_3-1)-1,8)+1;
        temp = quanti_mat_DCT(index_1,index_2,index_3);
        temp = Mydec2bin(temp);
        temp(length(temp)) = info_bin(i,j);
        quanti_mat_DCT(index_1,index_2,index_3) = Mybin2dec(temp);
    end
end
for i = 1 : 1 : block_num
    quanti_mat(:,i) =
[quanti_mat_DCT(1,1,i);(ZigZag(quanti_mat_DCT(:,:,i))).'];
end
quanti_mat = int16(quanti_mat);
seq_DC = zeros(1,block_num);
seq_AC = zeros(block_num,63);
code_AC = [];
for i = 1 : 1 : block_num
    if i == 1
        seq_DC(i) = quanti_mat(1,i);
    else

```



```

        seq_DC(i) = quanti_mat(1,i-1)-quanti_mat(1,i);
    end
    seq_AC(i,:) = (quanti_mat(2:64,i)).';
end
code_DC = EncodeDC(seq_DC);
for i = 1 : 1 : block_num
    code_AC = [code_AC,EncodeAC(seq_AC(i,:))];
end
code_AC = str2num(code_AC(:))';
code_DC = str2num(code_DC(:))';
[dct_AC_seq] = DecodeAC(code_AC);
[dct_DC_seq] = DecodeDC(code_DC);
dct_AC_mat = zeros(8,8,block_num);
for i = 1 : 1 : block_num
    dct_AC_mat(:,:,i) = InverseZiZag(dct_AC_seq(63*(i-1)+1:63*i));
end
[dct_mat] = DctMatReform(dct_AC_mat,dct_DC_seq);
[image_DCT_1] = ImageReform(dct_mat,height,width);
imwrite(uint8(image_DCT_1),'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_3\\dct_1.jpg');
[proc_graph_1,blocks_1,block_num_1,height_1,width_1] =
GraphicsDivide(image_DCT_1-128);
for i = 1 : 1 : block_num
    dct_mat(:,:,i) = round(DCT(blocks_1(:,:,i))./QTAB);
end
info_trans = zeros(info_size(1),info_size(2));
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        num = info_size(2)*(i-1)+j;
        index_3 = floor((num-1)/64)+1;
        index_1 = floor((num - 64*(index_3-1)-1)/8)+1;
        index_2 = mod(num - 64*(index_3-1)-1,8)+1;
        list = [list,[index_1;index_2;index_3]];
        temp = dct_mat(index_1,index_2,index_3);
        temp = Mydec2bin(temp);
        info_trans(i,j) = temp(length(temp));
    end
end
rate = 0;
cnt = 0;
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        if info_trans(i,j) == info_bin(i,j)

```

```

        cnt = cnt + 1;
    end
end
end
pixel_num = 64*block_num;
MSE =
1/pixel_num*sum(sum((image_DCT_1-double(hall_gray)).*(image_DCT_1-double(hall_gray))));
PSNR = 10*log10(255^2/MSE);
display(PSNR);
rate = cnt/(info_size(1)*info_size(2));
display(rate);
info_trans = char(bin2dec(char(info_trans)))';
display(info_trans);

```

加密方式 2 脚本:

```

clear all,close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\hall.mat;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\JpegCoeff.mat;
hall_gray = double(hall_gray);
[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(hall_gray-128);
origin_info = 'signal and system is fun';
info_bin = dec2bin(double(origin_info));
info_size = size(info_bin);
quanti_mat_DCT = zeros(8,8,block_num);
quanti_mat = zeros(64,block_num);
for i = 1 : 1 : block_num
    quanti_mat_DCT(:,:,i) = round(DCT(blocks(:,:,i))./QTAB);
end
quanti_mat = int16(quanti_mat);
quanti_mat_DCT = int16(quanti_mat_DCT);
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        num = info_size(2)*(i-1)+j;
        temp = quanti_mat_DCT(1,1,num);
        temp = Mydec2bin(temp);
        temp(length(temp)) = info_bin(i,j);
        quanti_mat_DCT(1,1,num) = Mybin2dec(temp);
    end
end

```

```

        end
    end
    for i = 1 : 1 : block_num
        quanti_mat(:,i) =
        [quanti_mat_DCT(1,1,i);(ZigZag(quanti_mat_DCT(:,:,i))).'];
    end
    seq_DC = zeros(1,block_num);
    seq_AC = zeros(block_num,63);
    code_AC = [];
    for i = 1 : 1 : block_num
        if i == 1
            seq_DC(i) = quanti_mat(1,i);
        else
            seq_DC(i) = quanti_mat(1,i-1)-quanti_mat(1,i);
        end
        seq_AC(i,:) = (quanti_mat(2:64,i)).';
    end
    code_DC = EncodeDC(seq_DC);
    for i = 1 : 1 : block_num
        code_AC = [code_AC,EncodeAC(seq_AC(i,:))];
    end
    code_AC = str2num(code_AC(:))';
    code_DC = str2num(code_DC(:))';
    [dct_AC_seq] = DecodeAC(code_AC);
    [dct_DC_seq] = DecodeDC(code_DC);
    dct_AC_mat = zeros(8,8,block_num);
    for i = 1 : 1 : block_num
        dct_AC_mat(:,:,i) = InverseZiZag(dct_AC_seq(63*(i-1)+1:63*i));
    end
    [dct_mat] = DctMatReform(dct_AC_mat,dct_DC_seq);
    [image_DCT_2] = ImageReform(dct_mat,height,width);
    imwrite(uint8(image_DCT_2),'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\dct_2.jpg');
    [proc_graph_1,blocks_1,block_num_1,height_1,width_1] =
    GraphicsDivide(image_DCT_2-128);
    for i = 1 : 1 : block_num
        dct_mat(:,:,i) = round(DCT(blocks_1(:,:,i))./QTAB);
    end
    info_trans = zeros(info_size(1),info_size(2));
    info_trans = zeros(info_size(1),info_size(2));
    for i = 1 : 1 : info_size(1)
        for j = 1 : 1 : info_size(2)
            num = info_size(2)*(i-1)+j;

```

```

        temp = dct_mat(1,1,num);
        temp = Mydec2bin(temp);
        info_trans(i,j) = temp(length(temp));
    end
end
cnt = 0;
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        if info_trans(i,j) == info_bin(i,j)
            cnt = cnt + 1;
        end
    end
end
pixel_num = 64*block_num;
MSE =
1/pixel_num*sum(sum((image_DCT_2-double(hall_gray)).*(image_DCT_2-double(hall_gray))));
PSNR = 10*log10(255^2/MSE);
display(PSNR);
rate = cnt/(info_size(1)*info_size(2));
display(rate);
info_trans = char(bin2dec(char(info_trans)))';
display(info_trans);

```

加密方式 3 脚本:

```

clear all,close all, clc;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\hall.mat;
load C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\图像处理大作业\\图像处理所需资源\\JpegCoeff.mat;
hall_gray = double(hall_gray);
[proc_graph,blocks,block_num,height,width] =
GraphicsDivide(hall_gray-128);
origin_info = 'signal and system is fun';
info_bin = dec2bin(double(origin_info));
info_size = size(info_bin);
bin_num = info_size(1)*info_size(2);
temp = info_bin';
bin_seq = reshape(temp,[1,info_size(1)*info_size(2)]);
quanti_mat_DCT = zeros(8,8,block_num);
quanti_mat = zeros(64,block_num);

```

```

for i = 1 : 1 : block_num
    quanti_mat_DCT(:,:,i) = round(DCT(blocks(:,:,i))./QTAB);
    quanti_mat(:,i) =
[quanti_mat_DCT(1,1,i);(ZigZag(quanti_mat_DCT(:,:,i))).'];
end
quanti_mat = int16(quanti_mat);
bin_cnt = 1;
for i = 1 : 1 : block_num
    for j = 64 : -1 : 2
        if quanti_mat(j,i) ~= 0
            if j~= 64
                if bin_seq(bin_cnt) == '1'
                    quanti_mat(j+1,i) = 1;
                else
                    quanti_mat(j+1,i) = -1;
                end
            else
                if bin_seq(bin_cnt) == '1'
                    quanti_mat(j,i) = 1;
                else
                    quanti_mat(j,i) = -1;
                end
            end
            bin_cnt = bin_cnt + 1;
            break;
        end
    end
    if bin_cnt > bin_num
        break;
    end
end
seq_DC = zeros(1,block_num);
seq_AC = zeros(block_num,63);
code_AC = [];
for i = 1 : 1 : block_num
    if i == 1
        seq_DC(i) = quanti_mat(1,i);
    else
        seq_DC(i) = quanti_mat(1,i-1)-quanti_mat(1,i);
    end
    seq_AC(i,:) = (quanti_mat(2:64,i)).';
end
code_DC = EncodeDC(seq_DC);

```

```

for i = 1 : 1 : block_num
    code_AC = [code_AC,EncodeAC(seq_AC(i,:))];
end
code_AC = str2num(code_AC(:))';
code_DC = str2num(code_DC(:))';
[dct_AC_seq] = DecodeAC(code_AC);
[dct_DC_seq] = DecodeDC(code_DC);
dct_AC_mat = zeros(8,8,block_num);
for i = 1 : 1 : block_num
    dct_AC_mat(:,:,i) = InverseZiZag(dct_AC_seq(63*(i-1)+1:63*i));
end
[dct_mat] = DctMatReform(dct_AC_mat,dct_DC_seq);
[image_DCT_3] = ImageReform(dct_mat,height,width);
imwrite(uint8(image_DCT_3),'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_2\\dct_3.jpg');
[proc_graph_1,blocks_1,block_num_1,height_1,width_1] =
GraphicsDivide(image_DCT_3-128);
for i = 1 : 1 : block_num
    dct_mat(:,:,i) = round(DCT(blocks_1(:,:,i))./QTAB);
end
dct_AC_seq = [];
for i = 1 : 1 : block_num
    dct_AC_seq = [dct_AC_seq,ZigZag(dct_mat(:,:,i))];
end
trans_bin = zeros(1,bin_num);
bin_cnt = 1;
i = 1;
while bin_cnt <= bin_num
    for j = 63*i : -1 : 63*(i-1)+1
        if dct_AC_seq(j) == 1
            trans_bin(bin_cnt) = 1;
            bin_cnt = bin_cnt + 1;
            break;
        end
        if dct_AC_seq(j) == -1
            trans_bin(bin_cnt) = 0;
            bin_cnt = bin_cnt + 1;
            break;
        end
    end
    i = i + 1;
end
info_trans = zeros(info_size(1),info_size(2));

```

```

for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        info_trans(i,j) = char(trans_bin(info_size(2)*(i-1)+j)+48);
    end
end
cnt = 0;
for i = 1 : 1 : info_size(1)
    for j = 1 : 1 : info_size(2)
        if info_trans(i,j) == info_bin(i,j)
            cnt = cnt + 1;
        end
    end
end
pixel_num = 64*block_num;
MSE =
1/pixel_num*sum(sum((image_DCT_3-double(hall_gray)).*(image_DCT_3-double(hall_gray))));
PSNR = 10*log10(255^2/MSE);
display(PSNR);
rate = cnt/(info_size(1)*info_size(2));
display(rate);
info_trans = char(bin2dec(char(info_trans)))';
display(info_trans);

```

(3) 核心代码说明:

三种加密方式的基本流程均为先对原图像进行分割、DCT、量化。对量化后的系数进行嵌密。具体方式为：方法 1 将每一个 bit 位按照顺序依次取待到量化后的矩阵中每个元素的最后一个 bit 位。方法 2 将每一个 bit 按照顺序依次取待到量化后矩阵的 DC 系数最后一个 bit 位(本实验信息位够用, 如果不够用则按照 ZigZag 顺序依次替换 AC 系数的最低 bit 位)。方法 3 用±1 表示信息, 将信息为添加在 ZigZag 顺序最后一个非零 DCT 系数之后, 若最后一个系数不为零, 则用信息为替换该位。当 DCT 域加密后对加密的矩阵进行编码操作即可。

(4) 运行结果：

通过加密方式 1 嵌密后得到的图像：



通过加密方式 1 得到的 PSNR、正码率以及解密信息：

```
命令行窗口

PSNR =

    29.8529

rate =

    1

info_trans =

    'signal and system is fun'
```

通过加密方式 1 得到的 AC 码长：23362

通过加密方式 1 得到的 DC 码长：2054

通过加密方式 2 嵌密后得到的图像：



通过加密方式 2 得到的 PSNR、正码率以及解密信息：

```
>> PP_chapter3_2_2

PSNR =

    31.0886

rate =

    1

info_trans =

    'signal and system is fun'
```

通过加密方式 2 得到的 AC 码长:23073

通过加密方式 2 得到的 DC 码长:2064

通过加密方式 3 嵌密后得到的图像：



通过加密方式 3 得到的 PSNR、正码率以及解密信息：

```
命令行窗口

PSNR =

    30.1696

rate =

    1

info_trans =

    'signal and system is fun'
```

通过加密方式 3 得到的 AC 码长:23577

通过加密方式 3 得到的 DC 码长:2054

(5) 结果分析:

三种加密方式的压缩分别为 6.345、6.416、6.292。隐藏方法 1 与隐藏方法 3 由于都改变了 AC 系数矩阵, 导致连 0 的个数减少, 因此 AC 码长增加, 导致压缩比下降。隐藏方法 2 由于只改变了 DC 系数因此不会导致 AC 码长增加, 从而压缩比基本不变。三种方法的解密正确率均为 100%, 能够完全恢复出隐藏的信息。三种方法得到的加密图像 PSNR 分别为 29.59、31.09、29.92 均较高, 因此客观上加密的隐蔽性均较好, 但方式 1 明显感觉图片的左上角发生了变化因此在隐蔽性方面不如方法 2 和方法 3。

人脸检测:

4.1 给定的 Faces 目录下有 28 张人脸, 将其作为人脸训练标准 v

(a) 若样本人脸的大小不一样, 是否需要将图像调整为相同的大小

(b) 假设 L 的取值分别为 3、4、5, 所得的三个 v 之间有什么关系?

(1) 主要思想:

(a) 若人脸的大小不一样, 不需要将图像调整为相同的大小。因为人脸训练样本只需要提供各种颜色所占的比例, 相当于对不同大小的人脸进行了归一化, 因此不用要求人脸的大小一样。

(b) 当 $L = 4$ 对应的 v 的长度是 $L = 3$ 对应的 v 的长度的 8 倍。 $L = 5$ 对应的 v 的长度是 $L = 4$ 对应的 v 的长度的 8 倍。

4.2 设计一种能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地方加上红色边框)。选取一张多人图片, 对 L 取不同的值, 评价检测结果的区别。

(1) 主要思想:

先通过给定的 33 张人脸训练样本获得人脸标准向量。然后将待识别的彩色图像进行 8×8 分割, 按照图像块的顺序依次对每个 8×8 的图像块进行颜色判定。判定依据是统计该颜色块中各颜色所占比例, 得到该块的颜色向量。计算该块的颜色向量和人脸标准向量之间的距离。若二者之间的距离小于设定的阈值, 则将该块置为存在人脸的块。当所有的块都扫描结束以后, 使用 medfilt2 滤波。然后以块为单位, 按照顺序依次扫描每一个

块。当该块曾被置为存在人脸，则对该块附近进行广度优先搜索，认为该块的上下左右以及上左、上右、下左、下右块与之相连。如搜索得到的人脸块区域满足一定的大小和被认为是人脸的块满足一定的比例，则对该区域画红框，标记人脸。

(2) 核心代码：

标准人脸训练代码：

```
function [my_vector,temp] = FaceTraining()

    L = 3;
    %L = 4;
    %L = 5;
    my_vector = zeros(1,2^(3*L));
    temp = zeros(33,2^(3*L));
    for i = 1 : 1 : 33
        file_name = strcat('C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\
图像处理大作业\\图像处理所需资源\\Faces\\',int2str(i),'.bmp');
        face = imread(file_name,'bmp');
        my_vector = my_vector + SingleFaceTranning(face);
        temp(i,:) = SingleFaceTranning(face);
    end
    my_vector = my_vector/33;
end
```

8×8 块颜色统计函数 SingleFaceTraining:

```
function [chara_vector] = SingleFaceTranning(face)
% face training
    L = 3;
    %L = 4;
    %L = 5;
    face_size = size(face);
    u_R = zeros(1,2^(3*L));
    face = double(face);
    for i = 1 : 1 : face_size(1)
        for j = 1 : 1 : face_size(2)
            Rxy = floor(face(i,j,1)/(2^(8-L)));
            Gxy = floor(face(i,j,2)/(2^(8-L)));
            Bxy = floor(face(i,j,3)/(2^(8-L)));
            n = 2^(2*L)*Rxy+(2^L)*Gxy+Bxy+1;
            u_R(n) = u_R(n) + 1;
        end
    end
end
```

```

    chara_vector = 1/(face_size(1)*face_size(2))*u_R;
end

```

判断是否为人脸函数 IsFace:

```

function [face_flag] = IsFace(face_block,std_vector)
    threshold = 0.69;
    %threshold = 0.82;
    %threshold = 0.905;
    my_vector = SingleFaceTranning(face_block);
    distance = 1 - sqrt(my_vector)*(sqrt((std_vector).'));
    if distance <= threshold
        face_flag = 1;
    else
        face_flag = 0;
    end
end

```

画红框函数 RedBoundary:

```

function [recog_pic] = RedBoundary(detec_gray,proc_pic)
    MAX_FACE = 1000;
    queue = [];
    cursor_x = 0;
    cursor_y = 0;
    search_mode = 0;
    queue_top = 1;
    queue_end = 1;
    face_cnt = 0;
    [height,width] = size(detec_gray);
    visit_flag = zeros(height,width);
    red_boundary_mat = zeros(4,MAX_FACE);
    red_area_mat = zeros(1,MAX_FACE);
    face_size_mat = zeros(1,MAX_FACE);
    for i = 1 : 1 : height
        for j = 1 : 1 : width
            if (detec_gray(i,j) == 255)&&(~visit_flag(i,j))
                face_cnt = face_cnt + 1;
            end
        end
    end
end

```

%1 for visit
 %0 for not visit yet
 %dim 1: top limit
 %dim 2: bottom limit
 %dim 3: left limit
 %dim 4: right limit

```

        face_size_mat(face_cnt) = face_size_mat(face_cnt) + 1;
        red_boundary_mat(1:2,face_cnt) = [i;i];
        red_boundary_mat(3:4,face_cnt) = [j;j];
        search_mode = 1;
        queue = [i;j];
        queue_top = 1;
        queue_end = 2;
        visit_flag(i,j) = 1;
    end
    if search_mode == 1
        while queue_top ~= queue_end
            cursor_x = queue(1,queue_top);
            cursor_y = queue(2,queue_top);
            queue_top = queue_top + 1;
            for delta_x = -1 : 1 : 1
                for delta_y = -1 : 1 : 1
                    if (detec_gray(cursor_x + delta_x,cursor_y +
delta_y) == 255)&&(~visit_flag(cursor_x + delta_x,cursor_y + delta_y))
                        visit_flag(cursor_x + delta_x,cursor_y +
delta_y) = 1;

                        queue(1,queue_end) = cursor_x + delta_x;
                        queue(2,queue_end) = cursor_y + delta_y;
                        queue_end = queue_end + 1;
                        face_size_mat(face_cnt) =
face_size_mat(face_cnt) + 1;
                        red_boundary_mat(1,face_cnt) =
min(cursor_x+delta_x,red_boundary_mat(1,face_cnt));
                        red_boundary_mat(2,face_cnt) =
max(cursor_x+delta_x,red_boundary_mat(2,face_cnt));
                        red_boundary_mat(3,face_cnt) =
min(cursor_y+delta_y,red_boundary_mat(3,face_cnt));
                        red_boundary_mat(4,face_cnt) =
max(cursor_y+delta_y,red_boundary_mat(4,face_cnt));
                    end
                end
            end
        end
        red_area_mat(face_cnt) = (red_boundary_mat(2,face_cnt) -
red_boundary_mat(1,face_cnt) + 1)*(red_boundary_mat(4,face_cnt) -
red_boundary_mat(3,face_cnt) + 1);
        queue = [];
        queue_top = 1;
        queue_end = 1;
    end
end

```

```

        search_mode = 0;
    else
        continue;
    end
end
end
end
cnt_th = 0.25*max(face_size_mat);
ratio_th = 0.6;
lw_ratio_th = 0.55;
for i = 1 : 1 : face_cnt
    if(red_area_mat(i)/face_size_mat(i) >
ratio_th)&&(face_size_mat(i)>cnt_th)&&((red_boundary_mat(2,i)-red_bou
ndary_mat(1,i))/(red_boundary_mat(4,i)-red_boundary_mat(3,i))>lw_rati
o_th)

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(1,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(4,i),1) = 255;

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(1,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(4,i),2) = 0;

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(1,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(4,i),3) = 0;           %top
boundary

proc_pic(8*(red_boundary_mat(2,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(4,i),1) = 255;

proc_pic(8*(red_boundary_mat(2,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(4,i),2) = 0;

proc_pic(8*(red_boundary_mat(2,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(4,i),3) = 0;           %top
boundary

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(3,i),1) = 255;

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(3,i)-1)+1:8*red_boundary_mat(3,i),2) = 0;

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(2,i),8*(red

```

```

_boundary_mat(3,i)-1)+1:8*red_boundary_mat(3,i),3) = 0;           %left
boundary

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(4,i)-1)+1:8*red_boundary_mat(4,i),1) = 255;

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(4,i)-1)+1:8*red_boundary_mat(4,i),2) = 0;

proc_pic(8*(red_boundary_mat(1,i)-1)+1:8*red_boundary_mat(2,i),8*(red
_boundary_mat(4,i)-1)+1:8*red_boundary_mat(4,i),3) = 0;           %right
boundary
    end
end
    recog_pic = proc_pic;
end

```

人脸识别实现脚本：

```

prim_pic =
imread('C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_4\\new.
jpg');
[std_vector,temp] = FaceTraining();
[recog_pic,cnt] = FaceRecognition(prim_pic,std_vector);
imwrite(recog_pic,'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chap
ter_4\\19ans.jpg');

```

(3) 核心代码说明：

上述代码中两个向量之间的距离计算公式题目材料中所示。在 RedBoundary 函数中，判断一个区域是否为人脸的标准有以下 2 个：即在该区域中被认为是人脸的 block 数占该区域总的 block 数的比例需要大于 ratio_th。该区域的长宽之比需要大于下限阈值。

(4) 运行结果:

原始图像:



当 $L = 3$ 时, $\text{threshold} = 0.69$ 识别后的图像:



当 $L = 4$ 时, $\text{threshold} = 0.82$ 识别后的图像:



当 $L = 5$ 时, $\text{threshold} = 0.905$ 识别后的图像:



(5) 结果分析:

由运行结果可知,在不同的L以及适当的阈值下,人脸检测效果均较好,但是有少数人的手臂被当作了人脸。当L逐渐增大时,判断为人脸所需要的阈值 threshold 也随之增加。当选择的阈值比较合适时,能够比较准确的检测出人脸。

4.3 对上述所选多人图像进行以下处理

(a)顺时针旋转 90° (imrotate)

(b)保持高度不变,宽度拉伸为原来的两倍(imresize)

(c)适当改变颜色(imadjust)

对于不同的处理,人脸识别算法的检测结果如何?分析检测结果。

(1) 主要思想:

对原始图像进行不同的操作,在 L = 4, threshold = 0.855 的情况下检测人脸识别效果。

(2) 核心代码:

不同的处理后人脸识别脚本:

```
prim_pic =  
imread('C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_chapter_4\\new.  
jpg');  
pic_size = size(prim_pic);  
pic_rot = imrotate(prim_pic,-90);  
pic_resize = imresize(prim_pic,[pic_size(1),2*pic_size(2)]);  
pic_change_color = imadjust(prim_pic,[0.3,0.5],[0.8,1],1);  
[std_vector,temp] = FaceTraining();  
[recog_pic_rot,cnt] = FaceRecognition(pic_rot,std_vector);  
figure;  
imshow(recog_pic_rot);  
imwrite(recog_pic_rot,'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\PP_  
chapter_4\\rot.jpg');  
[recog_pic_resize,cnt] = FaceRecognition(pic_resize,std_vector);  
figure;  
imshow(recog_pic_resize);  
imwrite(recog_pic_resize,'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\  
PP_chapter_4\\resize.jpg');  
[recog_pic_rot,cnt] = FaceRecognition(pic_change_color,std_vector);  
figure;  
imshow(pic_change_color);
```

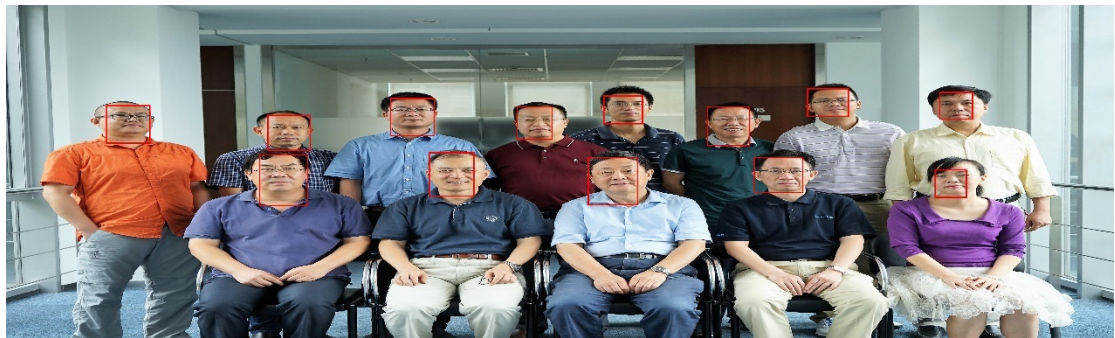
```
imwrite(pic_change_color,'C:\\Users\\HJSF\\Desktop\\Dynamic\\matlab\\  
PP_chapter_4\\change.jpg');
```

(3) 运行结果:

顺时针旋转 90°



保持高度不变，宽度拉伸为原来的两倍



适当改变颜色



(4) 结果分析:

由运行结果可知，当对图像进行旋转变换以及时将图像进行拉宽时，仍能正确识别出图片中的人脸。这是因为这两种方式并没有改变图像的颜色，并不影响基于颜色直方图的人脸识别算法。由于基于颜色直方图的人脸识别算法对于颜色有着很高的敏感性，因此适度改变颜色后将不能正确识别出人脸，需要采用其他方法(例如边缘轮廓检测等等)。

4.4 如果可以重新选择训练样本标准，你认为应该如何选择？

(1) 主要思想：

我认为如果可以重新选择训练样本，应该保证图片中基本全是人脸，并且最好是人脸的正部。同时对于不同肤色的人种，可以建立不同肤色人种的标准人脸训练样本，实现不同肤色人脸的识别。