

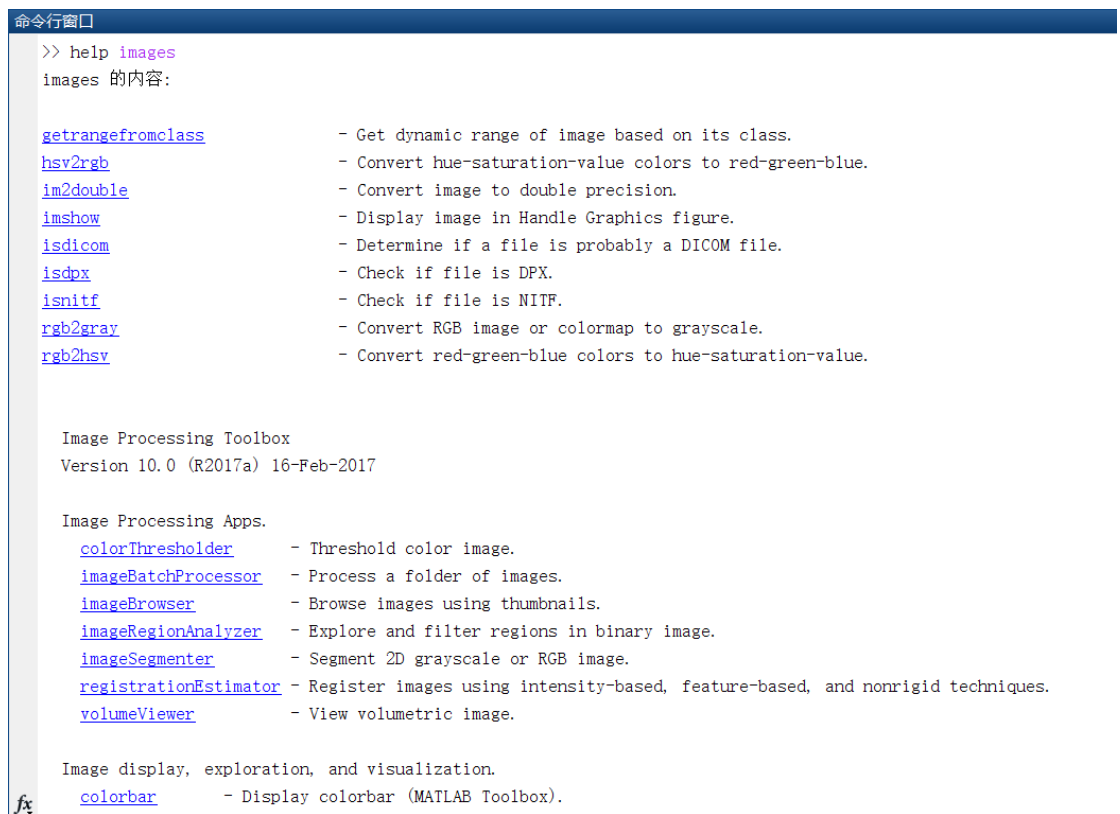
# MATLAB 图像处理大作业

漆耘含

无 63      2016011058

## 一、第一章基础知识练习题

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



```
命令窗口
>> help images
images 的内容:

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

Image Processing Toolbox
Version 10.0 (R2017a) 16-Feb-2017

Image Processing Apps.
colorThresholder - Threshold color image.
imageBatchProcessor - Process a folder of images.
imageBrowser     - Browse images using thumbnails.
imageRegionAnalyzer - Explore and filter regions in binary image.
imageSegmenter   - Segment 2D grayscale or RGB image.
registrationEstimator - Register images using intensity-based, feature-based, and nonrigid techniques.
volumeViewer     - View volumetric image.

Image display, exploration, and visualization.
colorbar         - Display colorbar (MATLAB Toolbox).
```

图 1 images 的部分函数

2. 利用 MATLAB 提供的 Image file I/O 函数分别完成一下处理：

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

原理：先通过 `size` 函数得到图像的尺寸，确定好中心，之后通过一个两重循环，如果点在圆内，则把该点的 Red 分量赋值为 255，把 Green 和 Blue 分量赋值为 0，效果如下：

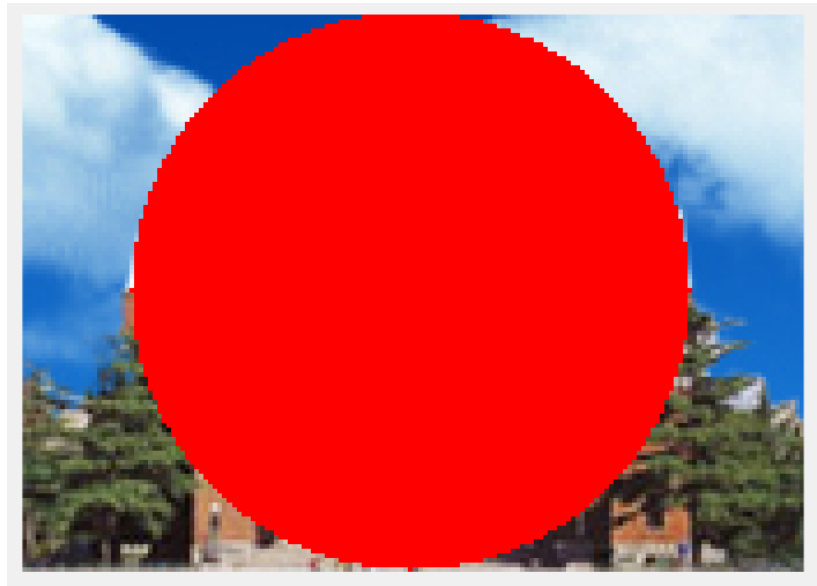


图 2 以宽为直径，中点为圆心画的圆

- 2) 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。

原理：将整张图划成一个个的格子，每一个格子有自己的编号，从左往右 1, 2, 3 编号，从上往下 1, 2, 3 编号，如果两个编号加起来是偶数，则涂黑；如果是奇数，则保留原图，效果如下：

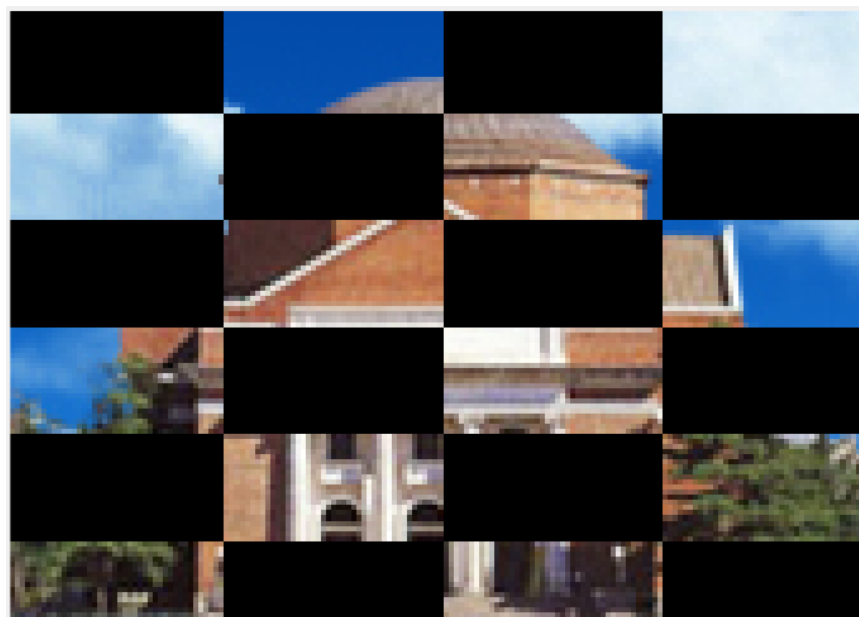


图 3 黑白格

## 二、 图像压缩编码练习题

1. 图像的预处理是将每个系那个素灰度值减去 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；

答：可以在变换域进行。第一种方法是在直接在原图像中减去 128 得到 cons，第二种方法是在变换域进行操作，再逆变换回来得到 inv，两者做差，找出其中的最大值 a，最终结果是  $a = 0.8527 \times 10^{-13}$ ，几乎为 0，则验证成功

```
代码：(image_21.m)

clear;

clc;

load('hall.mat');

G = hall_gray;

G1 = double(G(1:20,1:20));

elim = ones(20,20)*128;

%第一种方法

cons = G1 - elim;

%第二种方法

dct2_G1 = dct2(G1);

dct2_elim = dct2(elim);

inv = idct2(dct2_G1-dct2_elim);

%两者做差

sub = cons - inv;
```

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

原理：通过讲义中的方法，先得到矩阵 D，再通过  $D \cdot \text{image} \cdot D'$  得到 DCT，其中 D 为：

$$\sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \cdots & \sqrt{\frac{1}{2}} \\ \cos \frac{\pi}{2N} & \cos \frac{3\pi}{2N} & \cdots & \cos \frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(N-1)\pi}{2N} & \cos \frac{(N-1)3\pi}{2N} & \cdots & \cos \frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix}$$

得到自己实现的 trans1，再调用自带的函数 dct2()，得到 trans\_dct，两者做差进行比较，最终发现几乎完全相等，下面是 sub 的一部分：

```
>> sub
sub =
1.0e-11 *
1 至 15 列
      0    0.1393   -0.0036    0.0568   -0.0355   -0.0249    0.1080    0.1606    0.0035    0.1139    0.0079    0.0860   -0.1773    0.0347
      0.0400   -0.0085   -0.0028    0.0057    0.0071   -0.0094    0.0147   -0.0199    0.0002   -0.0064    0.0145   -0.0253   -0.0038    0.0135
     -0.0114   -0.0057    0.0071   -0.0028    0.0064   -0.0136   -0.0021   -0.0192    0.0032   -0.0188    0.0104   -0.0185    0.0092    0.0056
      0.0249    0.0071    0.0028    0.0057    0.0034   -0.0131   -0.0001   -0.0036    0.0043   -0.0055    0.0069   -0.0126    0.0050    0.0098
     -0.0313   -0.0028    0.0014   -0.0025    0.0010   -0.0091   -0.0014   -0.0078    0.0039   -0.0070    0.0018   -0.0047    0.0059    0.0065
      0.0021    0.0064    0.0021    0.0011    0.0014   -0.0064    0.0004   -0.0037    0.0002   -0.0036   -0.0020   -0.0010    0.0005    0.0047
      0.0895   -0.0018   -0.0006    0.0005    0.0018   -0.0011   -0.0018    0.0011    0.0007    0.0006   -0.0048    0.0032    0.0005    0.0034
      0.1581    0.0076    0.0026   -0.0020         0    0.0014    0.0004    0.0005   -0.0026    0.0034   -0.0001    0.0022    0.0006    0.0025
     -0.0033   -0.0011   -0.0007    0.0018    0.0007    0.0011   -0.0014    0.0020   -0.0017    0.0013   -0.0044    0.0058    0.0012   -0.0003
      0.1230    0.0051    0.0032   -0.0025    0.0007    0.0012   -0.0009    0.0030   -0.0015    0.0021   -0.0025    0.0027    0.0002    0.0002
      0.0304   -0.0056   -0.0010    0.0014         0    0.0014   -0.0020    0.0022    0.0011    0.0020   -0.0007    0.0021    0.0006    0.0000
      0.0868    0.0087    0.0034   -0.0059    0.0005    0.0004    0.0016    0.0013   -0.0014         0         0    0.0017   -0.0020    0.0008
     -0.2089    0.0001   -0.0006    0.0016    0.0007         0   -0.0018    0.0011   -0.0013    0.0012   -0.0002   -0.0041   -0.0004   -0.0013
      0.0339   -0.0013   -0.0001    0.0002    0.0005    0.0004   -0.0021    0.0012   -0.0003   -0.0004    0.0020   -0.0020    0.0006   -0.0008
      0.2366    0.0042    0.0011    0.0007   -0.0028    0.0021   -0.0024    0.0010   -0.0010    0.0003    0.0004   -0.0035   -0.0005   -0.0015
      0.4379    0.0122    0.0052   -0.0058   -0.0033    0.0023    0.0007    0.0012   -0.0020   -0.0004   -0.0004    0.0009   -0.0013    0.0003
      0.0310   -0.0009   -0.0012    0.0011    0.0002    0.0012    0.0008    0.0012   -0.0010   -0.0009    0.0023   -0.0032   -0.0000    0.0004
      0.0097   -0.0044   -0.0030    0.0014    0.0001   -0.0020    0.0000   -0.0018    0.0008   -0.0004    0.0015   -0.0036    0.0011   -0.0003
      0.1715    0.0042    0.0043   -0.0024   -0.0024    0.0027    0.0001    0.0015   -0.0014   -0.0018    0.0001    0.0001   -0.0008    0.0019
     -0.0911   -0.0006    0.0061   -0.0100    0.0040   -0.0015    0.0036   -0.0008   -0.0027   -0.0010   -0.0024   -0.0004    0.0014    0.0006
```

图 4 两者做差得到的 sub 矩阵

由图可以看出，量级在  $1e-11$ ，可以忽略不计，故编写成功。

```
代码：(image_22.m)

clear;

clc;

load('hall.mat');

G = hall_gray;

%选取一部分图像

G1 = double(G(1:20,1:20));
```

```

[width,length] = size(G1);
D = zeros(width,length);
%计算 D
for i = 1:width
    for j = 1:length
        if i == 1
            D(i,j) = sqrt(1/width);
        else
            D(i,j) = sqrt(2/width)*cos((i-1)*pi*(2*j-1)/(2*width));
        end
    end
end
%得到自己编写的 DCT
trans1 = D*G1*D';
%调用系统的函数
trans_dct = dct2(G1);
%两者做差
sub = trans1 - trans_dct;

```

3. 如果将 DCT 系数矩阵中右侧四列的系数全部置零，逆变换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？

原理：在计算 DCT 系数的时候，先分块（8\*8），然后计算 DCT，再将左侧或右侧的系数置零，结束之后再逆变换回去得到图像，如下所示：

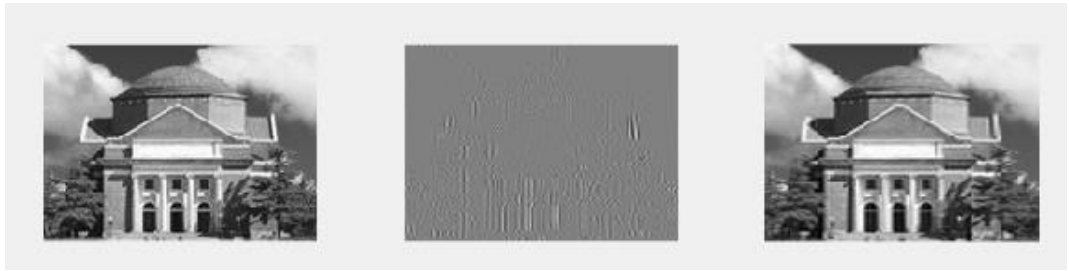


图 5 左图是原图，中图是左侧清零，右图是右侧清零

由图像可以看出，左侧清零后还原的图像几乎没有什么特征了，这是因为左侧清零，把主要的 DC 分量和主要的 AC 分量去掉了，故此几乎什么都没有；右侧和原图几乎差不多，不过在一些细节上有些微的差别，这是因为右边的都是一些细微的 AC 分量，去掉了对整体图像影响不大，只是一些细节地方有影响。

代码：(image\_23.m)

```
clear;
clc;
load('hall.mat');
G = double(hall_gray);
elim = double(ones(size(G))*128);
cons = G - elim;
[width,length] = size(cons);
image_left = zeros(size(cons));
image_right = zeros(size(cons));
for i = 1:width/8
    for j = 1:length/8
        %8*8 的一个小块
        test = cons((i-1)*8+1:i*8, (j-1)*8+1:j*8);
        %对原图进行 DCT 变化
        t_dct = dct2(test);
        %左侧清零
```

```

        t_dct_left = [zeros(8,4), t_dct(:, 5:8)];
        image_left((i-1)*8+1:i*8, (j-1)*8+1:j*8) =
idct2(t_dct_left);
        %右侧清零
        t_dct_right = [t_dct(:, 1:4), zeros(8,4)];
        image_right((i-1)*8+1:i*8, (j-1)*8+1:j*8) =
idct2(t_dct_right);
    end
end
figure
subplot(1,3,1), imshow(uint8(cons + elim));
subplot(1,3,2), imshow(uint8(image_left + elim));
subplot(1,3,3), imshow(uint8(image_right + elim));

```

4. 若对 DCT 系数分别作转置、旋转 90 度和旋转 180 度操作，逆变换后复原的图像有何变化，选取一块图验证你的结论；

原理：在上文中的代码稍微改动一下就可以了，改动的地方是分块计算 DCT 的时候，这里只列出变化部分的代码，效果图如下：

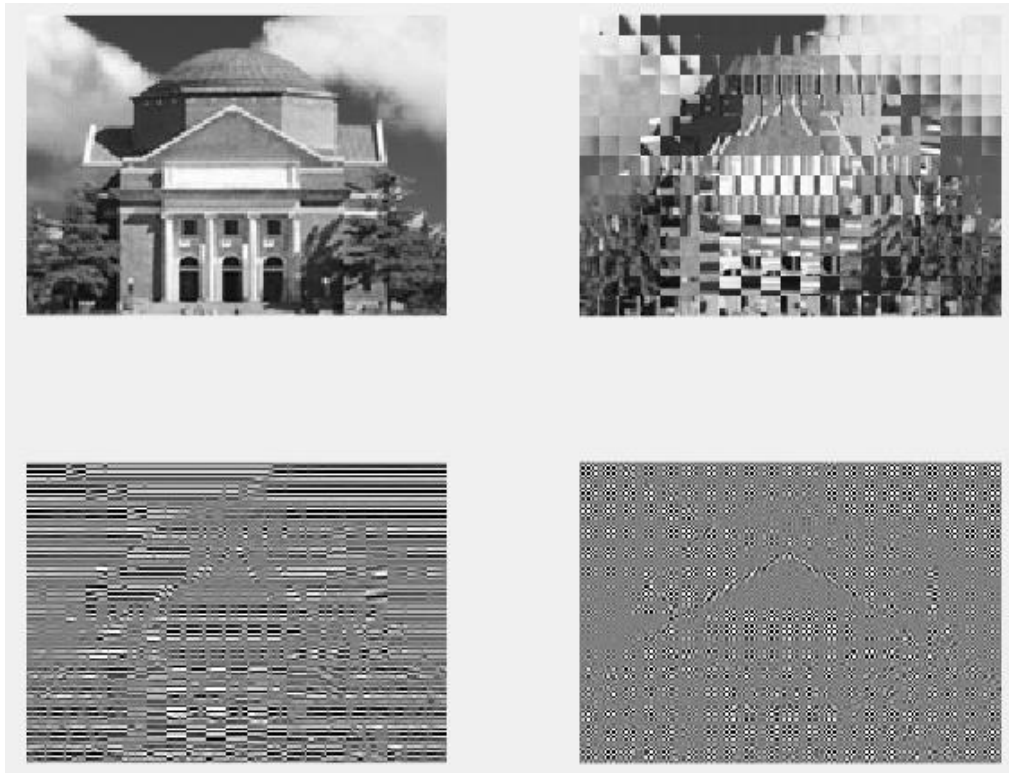


图 6 左上为原图，右上为转置，左下为旋转 90 度，右下为旋转 180 度

代码: (image\_24.m)

```
clear;
clc;
load('hall.mat');
G = double(hall_gray);
elim = double(ones(size(G))*128);
cons = G - elim;
[width,length] = size(cons);
image_trans = zeros(size(cons));
image_rot_90 = zeros(size(cons));
image_rot_180 = zeros(size(cons));
for i = 1:width/8
    for j = 1:length/8
        test = cons((i-1)*8+1:i*8, (j-1)*8+1:j*8);
        t_dct = dct2(test);
```



```

        %转置
        t_dct_trans = t_dct';
        image_trans((i-1)*8+1:i*8, (j-1)*8+1:j*8) =
idct2(t_dct_trans);
        %旋转 90 度
        t_dct_rot_90 = rot90(t_dct);
        image_rot_90((i-1)*8+1:i*8, (j-1)*8+1:j*8) =
idct2(t_dct_rot_90);
        %旋转 180 度
        t_dct_rot_180 = rot90(t_dct, 2);
        image_rot_180((i-1)*8+1:i*8, (j-1)*8+1:j*8) =
idct2(t_dct_rot_180);
    end
end
figure
subplot(2,2,1), imshow(uint8(cons + elim));
subplot(2,2,2), imshow(uint8(image_trans + elim));
subplot(2,2,3), imshow(uint8(image_rot_90 + elim));
subplot(2,2,4), imshow(uint8(image_rot_180 + elim));

```

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，说明它是一个高通（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 系数的高频频率分量更多。

原理：差分编码的递推式为：

$$\hat{c}_D(n) = \begin{cases} \bar{c}_D(n) & n = 1; \\ \bar{c}_D(n-1) - \bar{c}_D(n) & \text{elsewhere} \end{cases}$$

通过 freq() 函数可以得到频响：

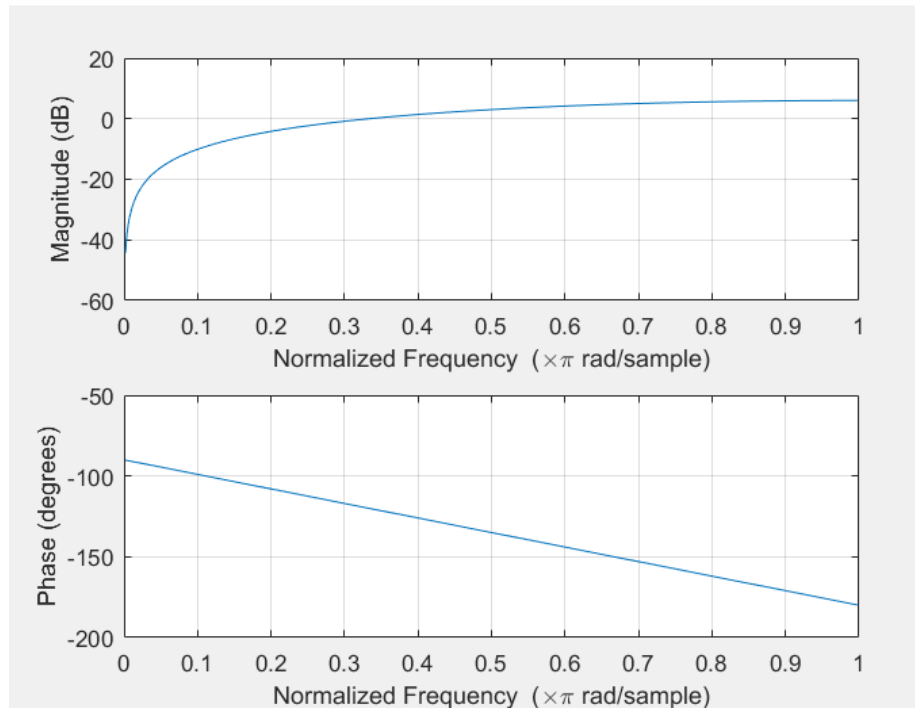


图 7 差分系统的频率响应

代码: (image\_25.m)

```
clear;
clc;

a = 1;
b = [-1, 1];

figure;
freqz(b, a);
```

6. DC 预测误差的取值和 Category 值有何关系? 如果利用预测误差计算其 Category?

答: DC 预测误差的取值  $k$  和 category 的值  $n$  之间满足以下关系:

$$2^{n-1} \leq k \leq 2^n$$

因此可以根据  $K$  的值计算出  $n$ :

$$n = \log_2 k + 1$$

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

原理: 因为是对一个  $8 \times 8$  的矩阵进行扫描, 最简单的方法就是直接把对应的矩阵元素写出来, 代码如下:

```
代码: (image_27.m)

%直接用标号, 因为是  $8 \times 8$ , 所以这样效率比较高

clear;

clc;

load('hall.mat');

G = double(hall_gray);

elim = double(ones(size(G))*128);

cons = G - elim;

test = cons(1:8,1:8);

matrix_test = dct2(test);

matrix_zig =
[1, 2, 9, 17, 10, 3, 4, 11, 18, 25, 33, 26, 19, 12, 5, 6, 13, 20, 27, 34, 41, ...
49, 42, 35, 28, 21, 14, 7, 8, 15, 22, 29, 36, 43, 50, 57, 58, 51, 44, 37, ...
30, 23, 16, 24, 31, 38, 45, 52, 59, 60, 53, 46, 39, 32, 40, 47, ...
54, 61, 62, 55, 48, 56, 63, 64];

a = reshape(matrix_test,1,64);

b = a(matrix_zig);
```

这样就简单有效地进行了 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 扫描后形成的列向量，第一行为各个块的 DC 系数；

原理：在分块进行 DCT 的时候，得到 DCT 系数之后点除量化矩阵，得到量化后的矩阵进行 Zig-Zag 扫描，然后 reshape 成一个列向量，保存在矩阵中。

```
代码：(image_28.m)

clear;

clc;

load('JpegCoeff.mat');
load('hall.mat');

Q = QTAB;

G = double(hall_gray);

elim = double(ones(size(G))*128);

cons = G - 128;

[width,length] = size(cons);

num_block = width*length/64;

out = zeros(64,num_block);

line = 1;

for i = 1:width/8

    for j = 1:length/8

        test = cons((i-1)*8+1:i*8, (j-1)*8+1:j*8);

        t_dct = dct2(test);

        %进行量化和扫描

        t_dct_lianghua = zig_zag(round(t_dct./Q));
```

```

        %将量化和扫描之后的保存在矩阵的一列中
        out(:, line) = t_dct_lianghua';
        line = line +1;
    end
end

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

原理：先对矩阵第一行（DC）进行编码，之后再对 AC 系数进行编码，编码方式为讲义中提及的，这里不再赘述，最终把得到的 `DC_code`、`AC_code`、`width` 和 `length` 写入文件中保存。

代码: (image\_29.m)

```
clear;

clc;

load('JpegCoeff.mat');
load('hall.mat');

Q = QTAB;

G = double(hall_gray);

elim = double(ones(size(G))*128);

cons = G - elim;

[width,length] = size(cons);

num_block = width*length/64;

out = zeros(64,num_block);

line = 1;

%进行 DCT、量化、扫描后擦偶存在 out 矩阵中

for i = 1:width/8

    for j = 1:length/8

        test = cons((i-1)*8+1:i*8, (j-1)*8+1:j*8);

        t_dct = dct2(test);

        t_dct_lianghua = zig_zag(round(t_dct./Q));

        out(:,line) = reshape(t_dct_lianghua,64,1);

        line = line +1;

    end

end

%DC 编码, 先进行差分

a = 1;

b = [-1,1];

DC = out(1,:);

DC_t = filter(b,a,DC);
```

```

DC_t(1) = DC(1);
DC_catg = floor(1+log2(abs(DC_t)));
DC_catg(DC_catg == -Inf) = 0;
DC_code = [];
%进行编码
for i = 1:num_block
    len = DCTAB(DC_catg(i)+1,1);%第一列 L
    DC_code = [DC_code,
DCTAB(DC_catg(i)+1,2:1+len),dec_cvt_bin(DC_t(i))];
end

%AC 编码
%两个常量
EOB = [1,0,1,0];
ZRL = [1,1,1,1,1,1,1,1,0,0,1];
AC_code = [];
for i = 1:num_block
    AC = out(2:64,i);
    Run = 0;
    Num_of_ZRL = 0;
    for j = 1:63
        if AC(j) == 0
            Run = Run +1;
            if Run == 16
                Num_of_ZRL =1;
                Run = 0;
            end
        else
            while Num_of_ZRL > 0

```

```

        AC_code = [AC_code, ZRL];
        Num_of_ZRL = Num_of_ZRL -1;
    end
    size = floor(log2(abs(AC(j))))+1;
    len = ACTAB(Run*10+size,3);
    AC_code = [AC_code,
ACTAB(Run*10+size,4:3+len),dec_cvt_bin(AC(j))];
    Run = 0;
end
end
AC_code = [AC_code,EOB];
end
legth = length;
%保存在文件中
save('Jpegcodes.mat','DC_code','AC_code','width','legth'
);

```

%编写的十进制到二进制的转换，因为这里牵涉到负数

```

function bin = dec_cvt_bin(a)
bit = floor(log2(abs(a)))+1;
bit(bit == -Inf) =1;
bin = zeros(1,bit);
remain = abs(a);
num =0;
while remain ~=0
    bin(bit-num) = mod(remain,2);
    remain = floor(remain/2);
    num = num+1;
end

```



```

    if a < 0
        bin = ~bin;
    end
end

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

```

#### 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;

原理: 压缩比=输入文件长度/输出码流长度, 计算得出 rate=6.4150。

```

代码: (image_210.m)
%压缩比
clear;
clc;
load('Jpegcodes.mat');
image = 8*length*width;

```

```

    after_process = length(AC_code)+ length(DC_code) +
length(dec_cvt_bin(width))+ length(dec_cvt_bin(legth));

    rate = image/after_process;
    %十进制转二进制函数
    function code = dec_cvt_bin(data)
    data_abs = abs(data);
    code_tmp = dec2bin(data_abs);
    if(data >= 0)
        code = code_tmp;
    else
        code = ~code_tmp;
        code = code +1;
    end
end

```

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

原理：解码的方法和编码的方式相反，具体的操作在讲义上有，这里不再赘述，解码后得到的图像如下：



图 8 左边是原图，右边是解码图

从图中可以看出，两张图大体上是一样的，不过有一些很小的细节有些

微出入。

MSE = 49.5873

PSNR = 31.1771

代码: (image\_211.m)

```
clear;
clc;
load('hall.mat');
load('JpegCoeff.mat');
load('Jpegcodes.mat');
flag = 1;
DC_decode = [];
while flag <= length(DC_code)
    for i = 1:12
        len = DCTAB(i,1);
        if flag-1+len <= length(DC_code) &&
any(DC_code(flag:flag-1+len)-DCTAB(i,2:1+len))==0
            if i == 1
                bin = DC_code(flag+len);
                flag = flag +1+len;
            else
                bin = DC_code(flag+len:flag+len-2+i);
                flag = flag-1+i+len;
            end
            %decode
            DC_decode = [DC_decode,bin_cvt_dec(bin,i)];
            break;
        end
    end
end
```

```

        end
    end

    for i =2:length(DC_decode)
        DC_decode(i) = DC_decode(i-1) - DC_decode(i);
    end

    %AC 部分
    flag = 1;
    num_of_block = 1;
    EOB = [1, 0, 1, 0];
    ZRL = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1];
    AC_decode = zeros(63, floor(width/8)*floor(legth/8));
    AC_decode_m = [];
    while flag <= length(AC_code)
        if      flag+10      <=      length(AC_code)      &&
any(AC_code(flag:flag+10) - ZRL) == 0
            AC_decode_m = [AC_decode_m, zeros(1, 16)];
            flag = flag +11;
        elseif      flag+3      <=      length(AC_code)      &&
any(AC_code(flag:flag+3)-EOB) == 0
            AC_decode(1:length(AC_decode_m), num_of_block) =
AC_decode_m';
            AC_decode_m = [];
            num_of_block = num_of_block +1;
            flag =flag +4;
        else
            for i = 1:160
                len = ACTAB(i, 3);
                if      flag-1+len      <=      length(AC_code)      &&

```

```

any(AC_code(flag:flag-1+len)-ACTAB(i,4:3+len)) == 0

        Run = ACTAB(i,1);
        size = ACTAB(i,2);
        bin = AC_code(flag+len:flag+len-1+size);
        AC_decode_m                                =
[AC_decode_m,zeros(1,Run),bin_cvt_dec(bin,size+1)];
        flag = flag+len+size;
        break;
    end
end
end
end

decode = [DC_decode;AC_decode];
img = zeros(width,legth);
num_o_b = 1;
for i = 1:(width/8)
    for j = 1:(legth/8)
        b = decode(:,num_o_b)';
        b = reshape(b,8,8);
        b_v = inv_zigzag(b).*QTAB;
        image((i-1)*8+1:i*8,(j-1)*8+1:j*8) = idct2(b_v);
        num_o_b = num_o_b+1;
    end
end

figure
subplot(1,2,1),imshow(uint8(hall_gray));
subplot(1,2,2),imshow(uint8(image+128));

```

```

MSE      =      1/width/length*sum(sum((double(image+128)-
double(hall_gray)).^2));

PSNR = 10*log10(255^2/MSE);

function r = inv_zigzag(b)
matrix_zig = [1,2,6,7,15,16,28,29;
               3,5,8,14,17,27,30,43;
               4,9,13,18,26,31,42,44;
               10,12,19,25,32,41,45,54;
               11,20,24,33,40,46,53,55;
               21,23,34,39,47,52,56,61;
               22,35,38,48,51,57,60,62;
               36,37,49,50,58,59,63,64];

r = b(matrix_zig);
end

function dec = bin_cvt_dec(bin,i)
if bin(1) == 0
    if i == 1
        bin_inv = [];
        dec = 0;
    else
        bin_inv = ~bin;
        dec = -1;
    end
else
    bin_inv = bin;
    dec = 1;
end

```

```

end
N = length(bin_inv);
dec_m = 0;
for i = 1:N
    dec_m = dec_m + bin_inv(i)*2^(N-i);
end
dec = dec*dec_m;
end

```

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。

原理：量化步长减小为原来的一半，即  $QTAB = QTAB./2$ ，得到的效果图如下：



图 9 左边是原图，右边是解码图

从图中可以看出，两张图依旧是差不多的，不过在细节上比量化步长大的要好一些。

压缩比 = 4.4

MSE = 24.6739

PSNR = 34.2084

代码 1: (image\_212\_1.m)

```
clear;

clc;

load('JpegCoeff.mat');
load('hall.mat');

Q = QTAB./2;

G = double(hall_gray);

elim = double(ones(size(G))*128);

cons = G - elim;

[width,length] = size(cons);

num_block = width*length/64;

out = zeros(64,num_block);

line = 1;

for i = 1:width/8
    for j = 1:length/8
        test = cons((i-1)*8+1:i*8, (j-1)*8+1:j*8);
        t_dct = dct2(test);
        t_dct_lianghua = zig_zag(round(t_dct./Q));
        out(:,line) = reshape(t_dct_lianghua,64,1);
        line = line +1;
    end
end

%DC

a = 1;

b = [-1,1];

DC = out(1,:);

DC_t = filter(b,a,DC);

DC_t(1) = DC(1);
```



```

DC_catg = floor(1+log2(abs(DC_t)));
DC_catg(DC_catg == -Inf) = 0;
DC_code = [];
for i = 1:num_block
    len = DCTAB(DC_catg(i)+1,1);%第一列 L
    DC_code = [DC_code,
DCTAB(DC_catg(i)+1,2:1+len),dec_cvt_bin(DC_t(i))];
end

%AC
EOB = [1,0,1,0];
ZRL = [1,1,1,1,1,1,1,1,1,0,0,1];
AC_code = [];
for i = 1:num_block
    AC = out(2:64,i);
    Run = 0;
    Num_of_ZRL = 0;
    for j = 1:63
        if AC(j) == 0
            Run = Run +1;
            if Run == 16
                Num_of_ZRL =1;
                Run = 0;
            end
        else
            while Num_of_ZRL > 0
                AC_code = [AC_code, ZRL];
                Num_of_ZRL = Num_of_ZRL -1;
            end
        end
    end
end

```

```

        size = floor(log2(abs(AC(j))))+1;
        len = ACTAB(Run*10+size, 3);
        AC_code = [AC_code,
ACTAB(Run*10+size, 4:3+len), dec_cvt_bin(AC(j))];
        Run = 0;
    end
end
AC_code = [AC_code, EOB];
end
length = length;
save('Jpegcodes_1.mat', 'DC_code', 'AC_code', 'width', 'length');

%十进制转二进制函数
function bin = dec_cvt_bin(a)
bit = floor(log2(abs(a)))+1;
bit(bit == -Inf) = 1;
bin = zeros(1, bit);
remain = abs(a);
num = 0;
while remain ~= 0
    bin(bit-num) = mod(remain, 2);
    remain = floor(remain/2);
    num = num+1;
end
if a < 0
    bin = ~bin;
end
end

```

```

%扫描函数

function b = zig_zag(a)

matrix_zig =
[1, 2, 9, 17, 10, 3, 4, 11, 18, 25, 33, 26, 19, 12, 5, 6, 13, 20, 27, 34, 41, ...
49, 42, 35, 28, 21, 14, 7, 8, 15, 22, 29, 36, 43, 50, 57, 58, 51, 44, 37, ...
30, 23, 16, 24, 31, 38, 45, 52, 59, 60, 53, 46, 39, 32, 40, 47, ...
54, 61, 62, 55, 48, 56, 63, 64];

m = reshape(a', 1, 64);
b = m(matrix_zig);

end

```

代码 2: (image\_212\_2.m)

```

clear;

clc;

load('hall.mat');
load('JpegCoeff.mat');
load('Jpegcodes_1.mat');

flag = 1;
DC_decode = [];

image = 8*length*width;

after_process = length(AC_code)+ length(DC_code) +
length(dec_cvt_bin(width))+ length(dec_cvt_bin(length));

rate = image/after_process;

while flag <= length(DC_code)
    for i = 1:12
        len = DCTAB(i,1);
        if flag-1+len <= length(DC_code) &&

```

```

any(DC_code(flag:flag-1+len)-DCTAB(i,2:1+len))==0

    if i == 1
        bin = DC_code(flag+len);
        flag = flag +1+len;
    else
        bin = DC_code(flag+len:flag+len-2+i);
        flag = flag-1+i+len;
    end

    %decode
    DC_decode = [DC_decode, bin_cvt_dec(bin, i)];
    break;
end

end

end

for i =2:length(DC_decode)
    DC_decode(i) = DC_decode(i-1) - DC_decode(i);
end

%AC 部分
flag = 1;
num_of_block = 1;
EOB = [1, 0, 1, 0];
ZRL = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1];
AC_decode = zeros(63, floor(width/8)*floor(legth/8));
AC_decode_m = [];
while flag <= length(AC_code)
    if      flag+10      <=      length(AC_code)      &&
any(AC_code(flag:flag+10) - ZRL) == 0
        AC_decode_m = [AC_decode_m, zeros(1, 16)];

```

```

        flag = flag +11;

        elseif      flag+3      <=      length(AC_code)      &&
any(AC_code(flag:flag+3)-EOB) == 0

            AC_decode(1:length(AC_decode_m), num_of_block) =
AC_decode_m';

            AC_decode_m = [];

            num_of_block = num_of_block +1;

            flag =flag +4;

        else

            for i = 1:160

                len = ACTAB(i, 3);

                if      flag-1+len      <=      length(AC_code)      &&
any(AC_code(flag:flag-1+len)-ACTAB(i, 4:3+len)) == 0

                    Run = ACTAB(i, 1);

                    size = ACTAB(i, 2);

                    bin = AC_code(flag+len:flag+len-1+size);

                    AC_decode_m                                =

[AC_decode_m, zeros(1, Run), bin_cvt_dec(bin, size+1)];

                    flag = flag+len+size;

                    break;

                end

            end

        end

    end

end

end

decode = [DC_decode;AC_decode];

img = zeros(width, legth);

num_o_b = 1;

for i = 1:(width/8)

```

```

        for j = 1:(length/8)
            b = decode(:, num_o_b)';
            b = reshape(b, 8, 8);
            b_v = inv_zigzag(b).*(QTAB./2);
            image((i-1)*8+1:i*8, (j-1)*8+1:j*8) = idct2(b_v);
            num_o_b = num_o_b+1;
        end
    end

figure
subplot(1, 2, 1), imshow(uint8(hall_gray));
subplot(1, 2, 2), imshow(uint8(image+128));

%rate

%计算 MSE 和 PSNR

MSE      =      1/width/length*sum(sum((double(image+128)-
double(hall_gray)).^2));

PSNR = 10*log10(255^2/MSE);

%逆 Zig-Zag 扫描函数
function r = inv_zigzag(b)
matrix_zig = [1, 2, 6, 7, 15, 16, 28, 29;
               3, 5, 8, 14, 17, 27, 30, 43;
               4, 9, 13, 18, 26, 31, 42, 44;
               10, 12, 19, 25, 32, 41, 45, 54;
               11, 20, 24, 33, 40, 46, 53, 55;
               21, 23, 34, 39, 47, 52, 56, 61;
               22, 35, 38, 48, 51, 57, 60, 62;
               36, 37, 49, 50, 58, 59, 63, 64];

r = b(matrix_zig);

```

```

end
%二进制转十进制函数
function dec = bin_cvt_dec(bin,i)
if bin(1) == 0
    if i == 1
        bin_inv = [];
        dec = 0;
    else
        bin_inv = ~bin;
        dec = -1;
    end
else
    bin_inv = bin;
    dec = 1;
end
N = length(bin_inv);
dec_m = 0;
for i = 1:N
    dec_m = dec_m + bin_inv(i)*2^(N-i);
end
dec = dec*dec_m;
end
%十进制转二进制函数
function bin = dec_cvt_bin(a)
bit = floor(log2(abs(a)))+1;
bit(bit == -Inf) =1;
bin = zeros(1,bit);
remain = abs(a);
num =0;

```

```

while remain ~=0
    bin(bit-num) = mod(remain,2);
    remain = floor(remain/2);
    num = num+1;
end
if a < 0
    bin = ~bin;
end
end
end

```

13. 看电视时偶尔能看到美丽的雪花图像，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。

原理：将 12 问中的图像换成 snow 即可，这里不列出代码（见文件：image\_213\_1.m 和 image\_213\_2.m），效果图如下：

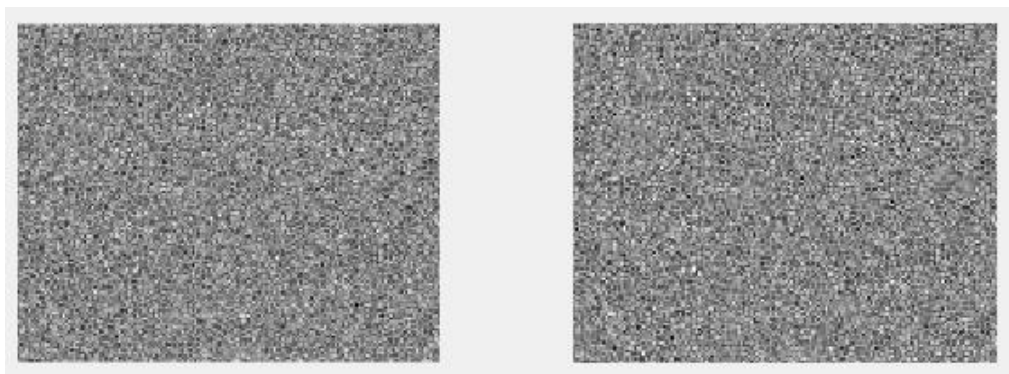


图 10 左图为原图，右图为编解码后的图

可以看出两者差别不大。

Rate = 3.6394

MSE = 331.5786

PSNR = 22.9259



### 三、 信息隐藏

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；

原理：将信息表示成二进制码流，依次用每位信息替换掉图像中各像素亮度分量的最低位，这里传递的信息是：handsome，效果图如下：



图 11 隐藏了信息的图像

通过直接解码得到 handsome，准确率为 100%

```
代码：(image_31.m)

clear;

clc;

load('hall.mat');

G = hall_gray;

secret = dec2bin(double('handsome'));

secret = [dec2bin(length(secret),7);secret];

[width,legth] = size(secret);

%写入信息

for i = 1:width

    for j = 1: legth

        m = dec2bin(G(i,j));

        m(length(m)) = secret(i,j);

        G(i,j) = uint8(bin2dec(m));
```

```

        end
    end
    imshow(G);
    save('mima.mat','G');
    %下面解码
    out = [];
    for i = 1:7
        m = dec2bin(G(1,i));
        m = m(length(m));
        out = [out m];
    end
    num = bin2dec(out);
    mima = zeros(num,7);
    jiema = zeros(1,num);
    for i = 2:num+1
        m_2 = [];
        for j = 1:7
            m = dec2bin(G(i,j));
            m_2 = [m_2 m(length(m))];
        end
        jiema(i-1) = bin2dec(m_2);
    end
    disp(char(jiema));

```

下面验证抗 JPEG 能力，对其进行编码和解码，得到的信息为：

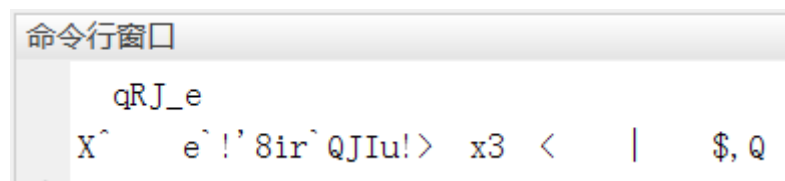


图 12 通过编解码后得到的信息

由图可见，信息已经完全变化了。

## 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 图像的质量变化以及压缩比变化；

### 1) 第一种方法

原理：用信息位逐一替换掉每个量化后的 DCT 系数的最低位，再进行熵编码。将信息转换成二进制码，reshape 为一个行向量，在进行量化并扫描之后得到 out 矩阵，将第一行中的系数转换成二进制码，然后依次替换二进制码的最低权重位，然后进行编码和解码，得到效果图（传递信息为 handsome is QYH）：



图 13 左侧为原图，右侧为隐含信息的图

从图中可以看出两者几乎完全相同，通过解码得到的信息为 'handsome is QYH'，准确率达 100%。

Rate = 6.4143

MSE = 50.2962

PSNR = 31.1154

关键代码 1（写入）：(image\_32\_1\_1.m)

%结束量化，开始写入信息

```
secret = dec2bin(double('handsome is QYH'));
```

```
secret = [dec2bin(length(secret),7);secret];
```

```

secret = str2num(secret(:));
secret = reshape(secret,1,16*7);
[w,1] = size(secret);

for i = 1: 1
    m = dec_cvt_bin(out(1,i));
    m(length(m)) = secret(i);
    out(1,i) = bin_cvt_dec(m,2);
end

```

关键代码（解码）：（image\_32\_1\_2.m）

```

re_msg = [];
for i = 1:L
    r = dec_cvt_bin(decode(1,i));
    re_msg = [re_msg,r(length(r))];
end

re_msg = reshape(re_msg,L/7,7);
msg = [];
for i = 2:L/7
    a = re_msg(i,:);
    a_dec = bin_cvt_dec(a,2);
    msg = [msg,a_dec];
end

msg_1 = char(msg);
disp(msg_1);

```

## 2) 第二种方法

原理：用信息位逐一替换掉若干量化后的 DCT 系数的最低位，再进行熵编码，注意不是每个 DCT 系数都嵌入了信息。基本方法和方法

一差不多，方法一是顺序替换，而方法二不是顺序替换，效果图如下：（传递的信息为:signal and system）



图 14 左侧为原图，右侧为隐含信息图

两者差不多，解码得出的信息是：‘signal and system’，准确率到达 100%。

Rate = 6.4148

MSE = 50.4556

PSNR = 31.1017

关键代码 1（写入）：（image\_32\_2\_1.m）

```
%结束量化，开始写入信息
str = 'signal and system';
secret = dec2bin(double(str));
secret = [dec2bin(length(secret),7);secret];
secret = str2num(secret(:));
secret = reshape(secret,1,(length(str)+1)*7);
[w,1] = size(secret);

for i = 1: 1
    m = dec_cvt_bin(out(1,i*2-1));
    m(length(m)) = secret(i);
    out(1,i*2-1) = bin_cvt_dec(m,2);
end
```

关键代码 2（解码）：（image\_32\_2\_2.m）

```

re_msg = [];
for i = 1:L
    r = dec_cvt_bin(decode(1, i*2-1));
    re_msg = [re_msg, r(length(r))];
end
re_msg = reshape(re_msg, L/7, 7);
msg = [];
for i = 2:L/7
    a = re_msg(i, :);
    a_dec = bin_cvt_dec(a, 2);
    msg = [msg, a_dec];
end
msg_1 = char(msg);
disp(msg_1);

```

### 3) 第三种方法

原理：先将带隐藏信息用 1, -1 的序列表示，再逐一将信息位追加在每个块 Zig-Zag 顺序的最后一个非零 DCT 系数之后，如果原本该图像的最后一个系数就不为零，那就用信息位替换该系数。这里用到的方法和前面两个方法有点出入，在写入信息的时候，在 Zig-Zag 扫描后保存在 out 矩阵的一列中，最后非零元素就是该列最后一个非零元素，直接在 out 矩阵中进行操作即可，下面是效果图：



图 15 左侧为原图，右侧为隐藏信息图

两个图差不多，解码所得信息为：‘1, 1, 1, -1, -1, 1, -1, 1, 1, 1, 1, 1, -1, -1, -1’，准确率是 100%

Rate = 6.4036

MSE = 49.9984

PSNR = 31.1412

关键代码 1（写入）：(image\_32\_3\_1.m)

%结束量化，开始写入信息

```
str = [1, 1, 1, -1, -1, 1, -1, 1, 1, 1, 1, 1, -1, -1, -1];
```

```
%secret = dec2bin(double(str));
```

```
%secret = [dec2bin(length(secret), 7); secret];
```

```
%secret = str2num(secret(:));
```

```
secret = str;
```

```
%secret = reshape(secret, 1, (length(str)+1)*7);
```

```
[w, 1] = size(secret);
```

```
for i = 1: 1
```

```
    m = out(:, i)';
```

```
    if m(length(m)) ~= 0
```

```
        m(length(m)) = secret(i);
```

```
    else
```

```
        for j = 1:length(m)-1
```

```
            if m(length(m)-j) ~= 0
```

```
                m(length(m)-j+1) = secret(i);
```

```
                break;
```

```
            end
```

```
        end
```

```
    end
```

```
    out(:, i) = m';
```

end
关键代码 2（解码）：(image_32_3_2.m) <pre> re_msg = []; for i = 1:L     m = decode(:, i)';     for j = 1:length(m)         if m(length(m)-j+1) ~= 0             re_msg = [re_msg, m(length(m)-j+1)];             break;         end     end end end disp(re_msg); </pre>

#### 四、 人脸检测

1. 所给资料 Faces 目录下包含 28 张人脸，试以其作为样本训练人脸标准  $v$

1) 样本人脸大小不一，是否需要首先将图像调整为相同大小？

答：不用大小相同，只需提取特征即可。

2) 假设  $L$  分别取 3, 4, 5，所得三个  $v$  之间有什么关系？

答：后者是前者的 8 倍。

3) 训练标准向量

原理如讲义中所述，训练结果如下：

$L = 3$ :



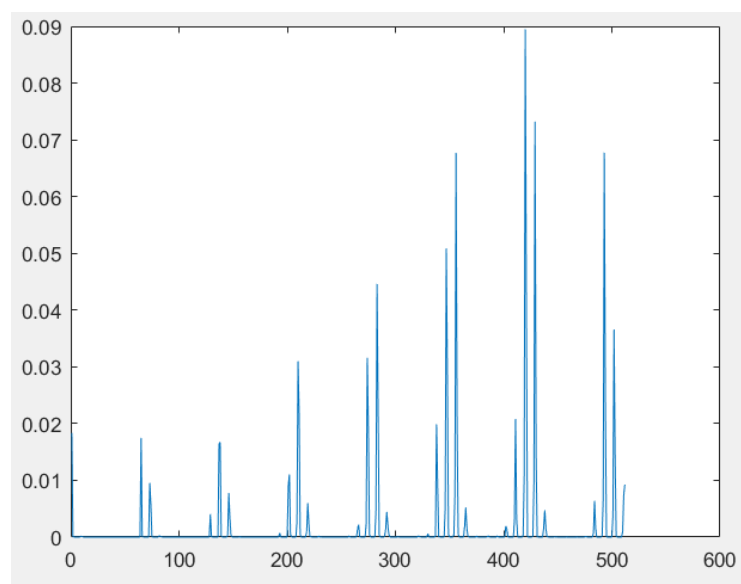


图 16  $L=3$

$L = 4$ ;

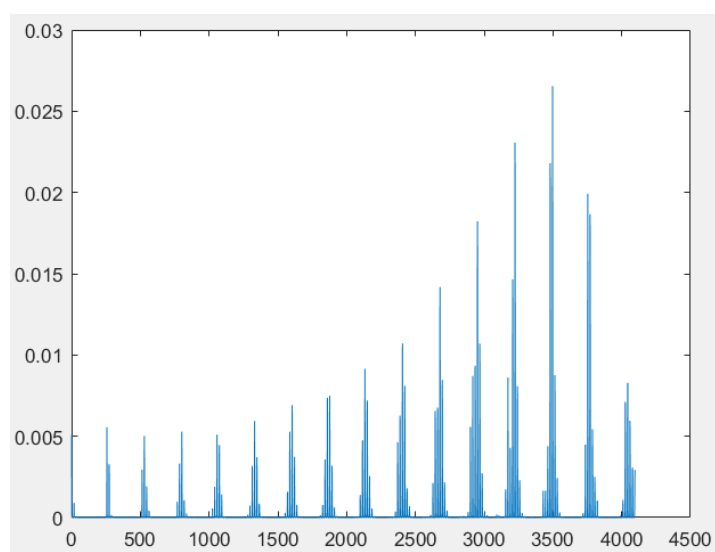


图 17  $L=4$

$L = 5$ ;

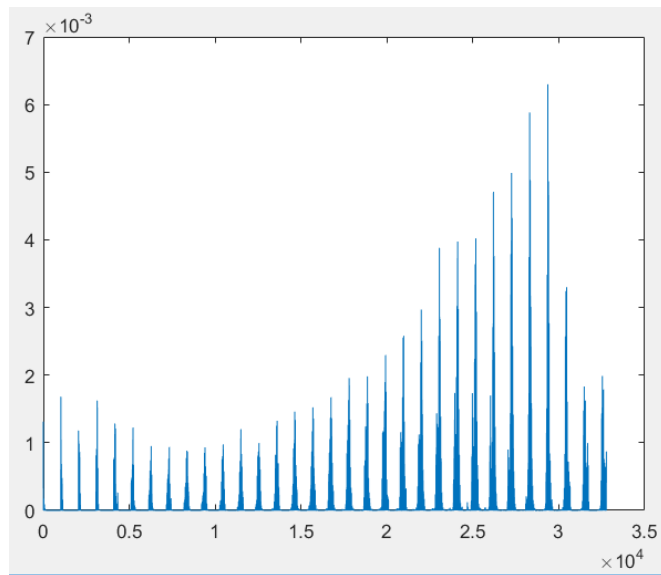


图 18 L=5

```

代码: (image_41.m)

clear;
clc;
L = 5;
len = 2^(3*L);
u_r = zeros(1, len);
freq = zeros(1, len);
R = zeros(1, len);
G = zeros(1, len);
B = zeros(1, len);
for i = 1:len
    bin = abs(dec2bin(i-1, 3*L)-48);
    R(i) = bin2dec(num2str(bin(1:L)));
    G(i) = bin2dec(num2str(bin(L+1:L*2)));
    B(i) = bin2dec(num2str(bin(2*L+1:L*3)));
end
for i = 1:33
    pic_addr = strcat('Faces\', int2str(i), '.bmp');

```

```

        %对当前的脸进行统计识别
        face = imread(pic_addr);
        face = floor(double(face)./(256/2^L));
        [H,W,D] = size(face);
        %进行统计
        for j = 1:len
            freq(j) = sum(sum(face(:, :, 1)==R(j) &
face(:, :, 2)==G(j) & face(:, :, 3)==B(j)));
        end
        freq = freq./H./W;
        u_r = u_r + freq;
    end
    u_r = u_r./33;
    %figure;
    %plot(u_r);
    u_r_5 = u_r;
    save('freq_5','u_r_5')

```

2. 设计一种从任意大小的图片中检测多张人脸的算法并变成实现（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，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。

原理：通过把一个图像分成很多小块，然后进行挨个匹配，当低于阈值的时候，则判断为是人脸，下面是例子：

$L=3$ ，阈值=0.25

原图：



检测后:



原图:



检测后图:



原图:

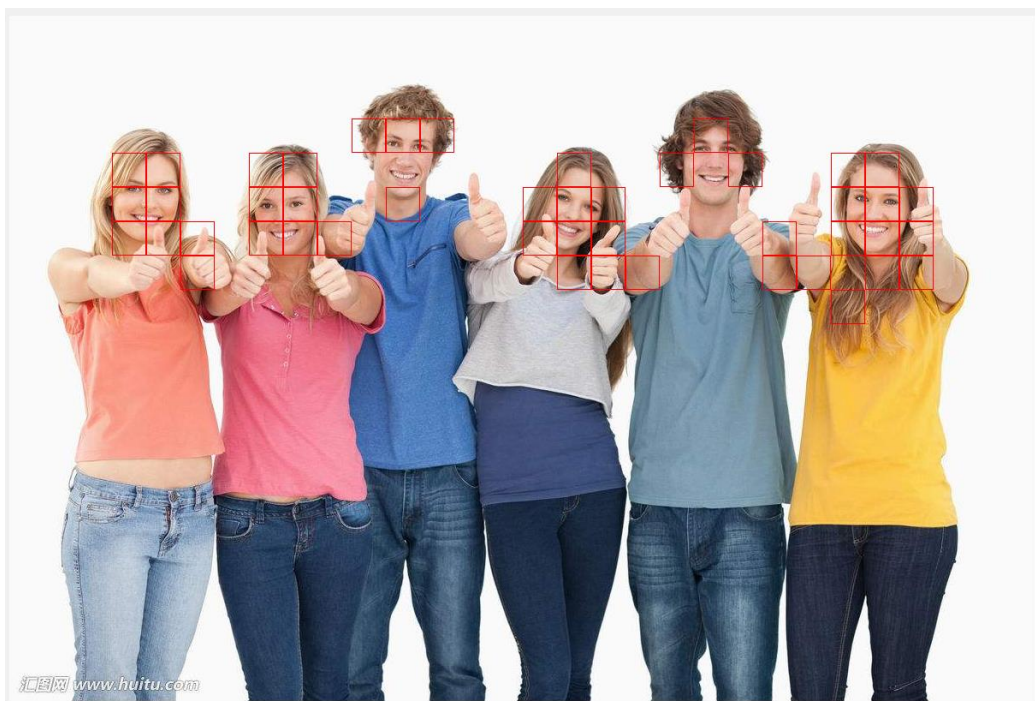




检测后:

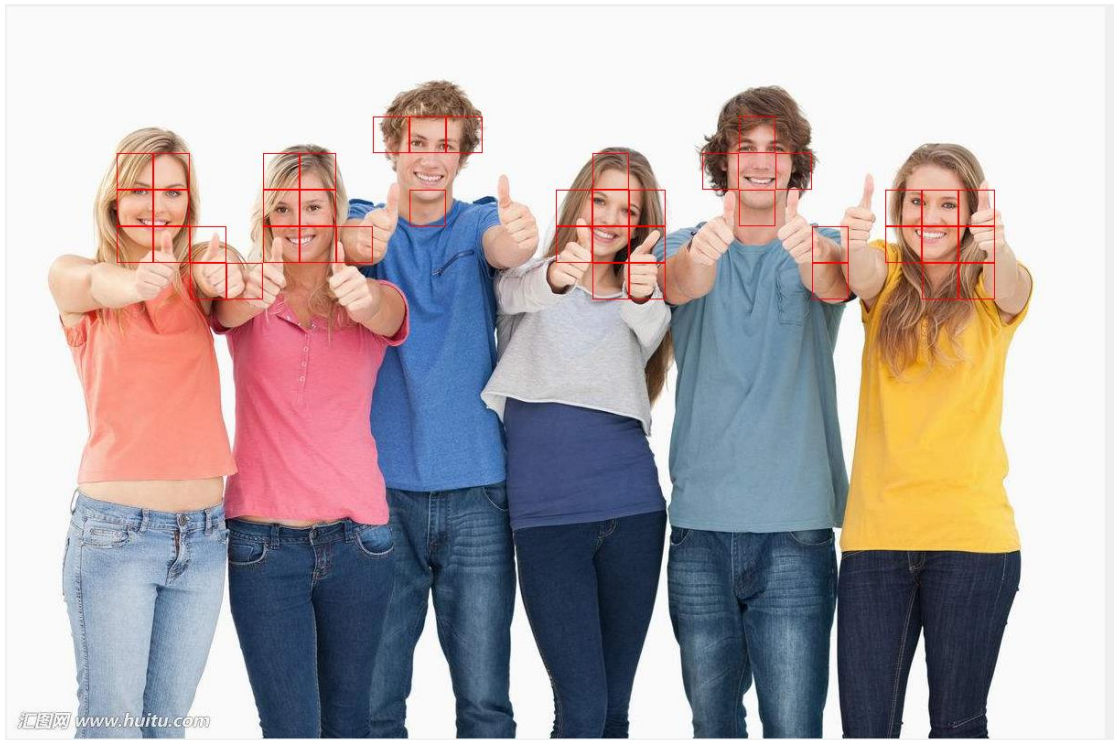


L=4, 阈值=0.45



L=5, 阈值=0.58





代码:

```
clear;
clc;
load('freq_5.mat');
test = imread('test3.bmp');
[H,W,D] = size(test);
v = 0.58;
L = 5;
len = 2^(3*L);
u_r = zeros(1,len);
freq = zeros(1,len);
R = zeros(1,len);
G = zeros(1,len);
B = zeros(1,len);
for i = 1:len
    bin = abs(dec2bin(i-1,3*L)-48);
```



```

        R(i) = bin2dec(num2str(bin(1:L)));
        G(i) = bin2dec(num2str(bin(L+1:L*2)));
        B(i) = bin2dec(num2str(bin(2*L+1:L*3)));
    end

    A = 40;
    C = 40;

    for i = 1:floor(H/A)
        for j = 1:floor(W/C)
            face = test((i-1)*A+1:i*A, (j-1)*C+1:j*C, 1:3);
            pre_face = face;
            face = floor(double(face)./(256/2^L));
            [H1,W1,D1] = size(face);
            for z = 1:len
                freq(z) = sum(sum(face(:, :, 1)==R(z) &
face(:, :, 2)==G(z) & face(:, :, 3)==B(z))));
            end
            freq = freq./H1./W1;
            %d = 1 - sum(sqrt(u_r_3(:).*freq(:)));
            d = 0;
            for h = 1:length(u_r_5)
                d = d + sqrt(u_r_5(h)*freq(h));
            end
            d = 1 - d;

            if d <= v

                face = pre_face;

```

```

        face(1, :, 1) = 255;
        face(:, 1, 1) = 255;
        face(H1, :, 1) = 255;
        face(:, H1, 1) = 255;
        face(1, :, 2) = 0;
        face(:, 1, 2) = 0;
        face(H1, :, 2) = 0;
        face(:, H1, 2) = 0;
        face(1, :, 3) = 0;
        face(:, 1, 3) = 0;
        face(H1, :, 3) = 0;
        face(:, H1, 3) = 0;

        test((i-1)*A+1:i*A, (j-1)*C+1:j*C, 1:3) =
face;

    end

end

end

imshow(test);

```

### 3. 对上述图像分别进行吐下处理后:

#### 1) 顺时针旋转 90 度

旋转 90 度: test = imrotate(test, 270);



2) 保持高度不变，宽度拉伸为原来的 2 倍

```
test = imresize(test, [H 2*W]);
```



3) 适当改变颜色

```
test = imadjust(test, [0.2 0.3 0; 0.8 0.6 1], []);
```



#### 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？

答：

在重新选择人脸样本进行训练的时候，因为最终是特点的平均值，所以如果有白人和黑人的话，这样特征就不是很明显，检测的结果会很不好，因此可以将肤色分开，白种人的训练一个特征向量出来，黑种人训练一个特征向量出来，分别进行识别，这样效果会更好。

其次，样本应该尽量选择多一点，这样人的面部特征也更明显。

### 五、大作业总结

通过这次大作业，初步了解了图像压缩、信息隐藏和人脸识别的知识。

在图像压缩部分，因为对原理不是很懂，所以导致进度特别慢，而且原理有时都会弄错，后来向同学请教了一下，慢慢弄懂原理，然后开始一点一点写代码，最后成功的做出了结果。在信息隐藏部分，没想到一张图中可以隐藏很多信息，这是之前没有考虑到的，也觉得特别有趣。在人脸识别部分，原理比较简单，但是检测出来的效果并不是很好，有时会把胳膊也当做人脸，还有在暖色调的图片中，检测效果极其的差，所以检测算法还有待提升。

这次大作业提升了我对 MATLAB 编程的能力和对 MATLAB 的理解，获益匪浅！.