

MATLAB 高级编程与工程应用

图像处理

学号：2020010768

班级：无 05

姓名：付宇辉

时间：2022 年 8 月 4 日

目录

一、实验名称与目的.....	2
二、实验内容.....	2
（一）基础知识.....	2
（二）图像压缩编码.....	4
（三）信息隐藏.....	14
（四）人脸检测.....	17
三、文件清单.....	21
四、心得体会.....	23
五、参考内容.....	24

一、实验名称与目的

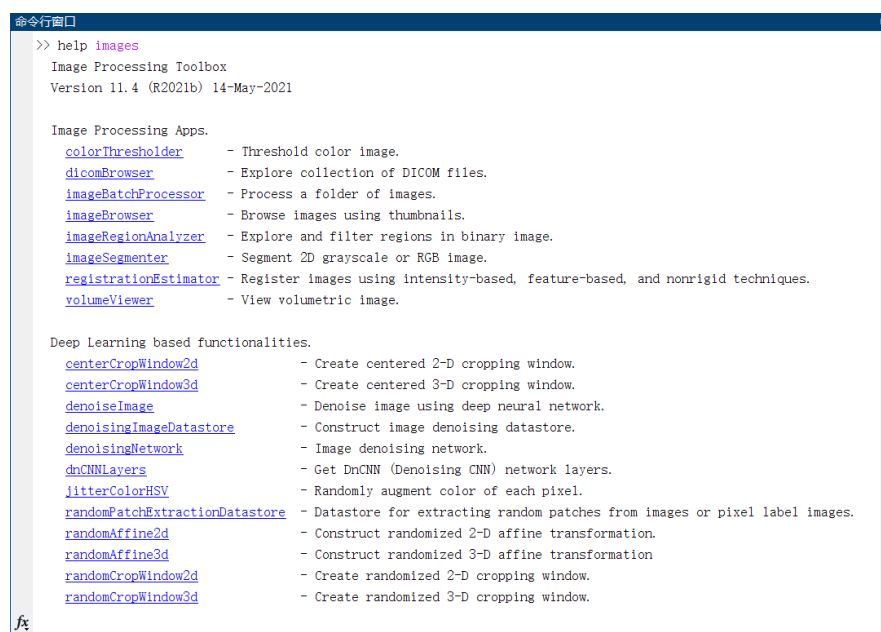
名称：图像处理

目的：了解计算机存储和处理图像的基础知识；练习 MATLAB 编程中矩阵运算等相关技巧，同时增进对二维信号及其频谱的物理意义的理解；了解在变换域进行信息隐藏的方法。

二、实验内容

（一）基础知识

1. 【题目描述】 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
Image Processing Toolbox
Version 11.4 (R2021b) 14-May-2021

Image Processing Apps.
  colorThresholder - Threshold color image.
  dicomBrowser     - Explore collection of DICOM files.
  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.

Deep Learning based functionalities.
  centerCropWindow2d - Create centered 2-D cropping window.
  centerCropWindow3d - Create centered 3-D cropping window.
  denoiseImage       - Denoise image using deep neural network.
  denoisingImageDatastore - Construct image denoising datastore.
  denoisingNetwork   - Image denoising network.
  dnCNNTLayers       - Get DnCNN (Denoising CNN) network layers.
  jitterColorHSV     - Randomly augment color of each pixel.
  randomPatchExtractionDatastore - Datastore for extracting random patches from images or pixel label images.
  randomAffine2d     - Construct randomized 2-D affine transformation.
  randomAffine3d     - Construct randomized 3-D affine transformation.
  randomCropWindow2d - Create randomized 2-D cropping window.
  randomCropWindow3d - Create randomized 3-D cropping window.
```

2. 【题目描述】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，图像的长和宽中的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。

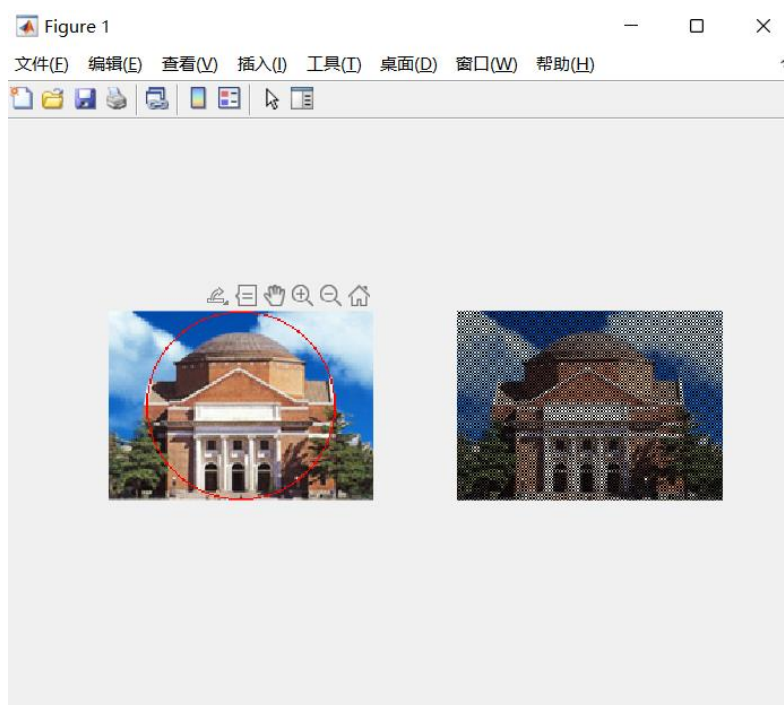
【主要思想】可以应用逻辑索引将图片需要修改数据的地方进行处理，从而进行对原图像的处理。

【核心代码】

```
% (1)
img_circle = hall_color;
radius = min(row/2, col/2);
dist = distance(row_idx , col_idx, (row+1)/2, (col+1)/2);
circle = (dist < radius^2) & (dist > 0.97 * radius^2);
img_circle(circle) = 255;
circle = ~circle;
img_circle(:, :, 2:3) = img_circle(:, :, 2:3) .* uint8(circle);
subplot(1,2,1)
imshow(img_circle);

% (2)
img_chessboard = hall_color;
mask = mod(row_idx + col_idx, 2); %可以快速实现提取网格信息。
img_chessboard = img_chessboard .* uint8(mask);
subplot(1,2,2)
imshow(img_chessboard);
```

【运行结果】



(二) 图像压缩编码

1. 【题目描述】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。

【主要思想】由于 DCT 具有线性性质，这个步骤可以在变换域进行。预处理将每个像素灰度值减去 128，再做 DCT，也可以将原图像直接进行 DCT 之后再与全为 128 的矩阵 DCT 相减，结果应该是一致的。

【核心代码】

```
c1 = dct2(hall_gray(1:8, 1:8) - 128);  
c2 = dct2(hall_gray(1:8, 1:8)) - dct2(zeros(8, 8) + 128);
```

【运行结果】将 c1, c2 展示之后的结果如下图

	1	2	3	4	5	6
1	919.7500	8.7800	-9.7500	2.1117	-4.0000	1.8
2	17.1450	5.7528	1.5922	2.1679	2.4500	0.3
3	10.8488	-9.6854	-1.0089	-2.0173	-1.7125	0.8
4	1.9176	5.5195	0.7804	0.4169	1.0319	2.1
5	-6.2500	-4.5243	2.3261	-1.2496	2	-0.2
6	-0.2737	1.1251	-2.4865	1.1016	1.1876	-1.8
7	-1.2465	1.2015	0.3750	-0.1294	0.4387	-2.3
8	0.6502	0.6545	-0.9012	-0.9573	-1.9742	0.9
9						
10						
11						
12						
13						
14						
15						
16						

【结果分析】两个变换域矩阵一致（误差很小），所以这个步骤是可以再在变换域进行的。

2. 【题目描述】请编程实现二维 DCT，并和 MATLAB 自带的函数库 `dct2` 比较是否一致。

【主要思想】根据大作业说明书的描述和公式，不难写出代码

【核心代码】

```
function C = my_dct2(P)
    N = size(P,1);
    D = (zeros([N - 1, N]) + [1 : 1 : N-1]') .* [1 : 2 : 2*N-1];
    D = sqrt(2 / N) * [zeros(1, N) + sqrt(1/2); cos(D * pi / (2*N))];
    C = D * double(P) * D';
end
```

【运行结果】同样利用 `hall_gray` 的左上角 8×8 块进行验证，将 `c1`, `c2` 展示之后的结果如下图

	1	2	3	4	5	6
1	919.7500	8.7800	-9.7500	2.1117	-4.0000	1.8
2	17.1450	5.7528	1.5922	2.1679	2.4500	0.3
3	10.8488	-9.6854	-1.0089	-2.0173	-1.7125	0.8
4	1.9176	5.5195	0.7804	0.4169	1.0319	2.1
5	-6.2500	-4.5243	2.3261	-1.2496	2	-0.2
6	-0.2737	1.1251	-2.4865	1.1016	1.1876	-1.8
7	-1.2465	1.2015	0.3750	-0.1294	0.4387	-2.3
8	0.6502	0.6545	-0.9012	-0.9573	-1.9742	0.9
9						
10						
11						
12						
13						
14						
15						
16						

	1	2	3	4	5	6
1	919.7500	8.7800	-9.7500	2.1117	-4.0000	1.8
2	17.1450	5.7528	1.5922	2.1679	2.4500	0.3
3	10.8488	-9.6854	-1.0089	-2.0173	-1.7125	0.8
4	1.9176	5.5195	0.7804	0.4169	1.0319	2.1
5	-6.2500	-4.5243	2.3261	-1.2496	2.0000	-0.2
6	-0.2737	1.1251	-2.4865	1.1016	1.1876	-1.8
7	-1.2465	1.2015	0.3750	-0.1294	0.4387	-2.3
8	0.6502	0.6545	-0.9012	-0.9573	-1.9742	0.9
9						
10						
11						
12						
13						
14						
15						
16						

【结果分析】可以看到自己编程实现的 DCT 与 MATLAB 自带库的 dct2 结果一致。

3. 【题目描述】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？

【主要思想】DCT 的系数矩阵中，左上角是直流低频分量，左下角是纵向变化的高频分量，右上角是横向变化的高频分量，右下角是横向和纵向变换的高频分量。所以将右侧四列置零，对图像整体影响不大，只是在横向的色彩变化比较模糊；但是将左侧四列置零，图像影响很大，直流和低频分量全部消失，同时图像纵向色彩变化会降低，但是横向变化上可能保留明显的变化，出现纵向纹理图像。

【核心代码】

```
clear_right_handle = @(block_struct) clear_right(block_struct.data);
clear_left_handle = @(block_struct) clear_left(block_struct.data);
c1 = blockproc(double(hall_gray)-128, [8 8], clear_right_handle);
c2 = blockproc(double(hall_gray)-128, [8 8], clear_left_handle);

clear_right_image = uint8(blockproc(c1, [8 8], @(block_struct)
idct2(block_struct.data) + 128));
clear_left_image = uint8(blockproc(c2, [8 8], @(block_struct)
idct2(block_struct.data) + 128));
```

【运行结果】将原始图片和进行左右列擦除后的图片 show 出来之后观察，图片如下图：



【结果分析】可以看到和分析结果一致。

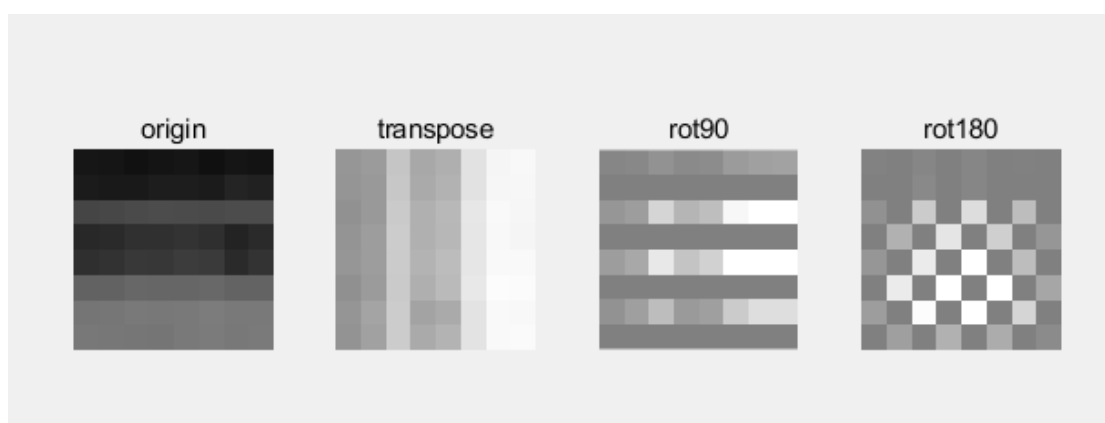
4. 【题目描述】若对 DCT 系数分别做转置、旋转 90 度和旋转 180 度操作(`rot90`), 逆变换后恢复的图像有何变化? 选取一块图验证你的结论。

【主要思想】做转置会使横纵向纹理较强和较弱的区块发生对调; 对于旋转 90 度, 会使低频分量的值变为纵向变化高频分量的值, 会出现较强的横向纹理; 矩阵旋转 180 度, 会使低频分量的值变为横纵高频分量的值, 出现大部分黑白变化的情况。

【核心代码】

```
c1 = dct2(sample)';  
transpose_image = uint8(idct2(c1)) + 128;  
subplot(1,4,2);  
imshow(transpose_image);  
title("transpose");
```

【运行结果】选取一个 8*8 的块, 运行结果如下图所示



【结果分析】可以看到和分析结果一致。

5. 【题目描述】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，说明他是一个__（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。

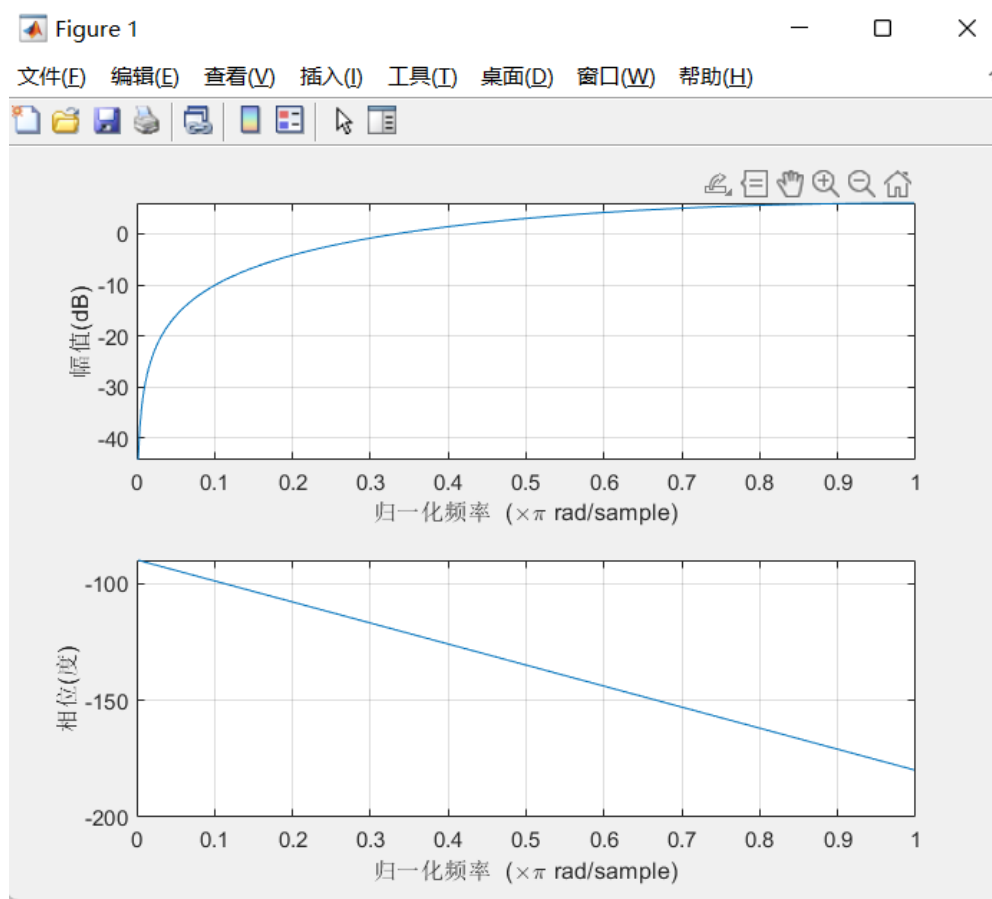
【主要思想】这个系统的差分方程为 $y[n] = x[n - 1] - x[n]$ ，利用 matlab 分析离散系统的工具，即可画出频率响应并进行相应的判断。

答案为：高通，高频

【核心代码】

```
a = 1;  
b = [-1 1];  
freqz(b, a);
```

【运行结果】运行结果如下图所示



【结果分析】可以看到和分析结果一致，答案为：高通，高频

6. 【题目描述】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？

【主要思想】假设 Category 的值为 c ，DC 预测误差的取值为 e ，则有：

$$c = \lceil \log_2(|e| + 1) \rceil$$

7. 【题目描述】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。

【主要思想】利用高级的矩阵索引方式可以实现 Zig-Zag 扫描

【核心代码】

```
function y = zig_zag(x)
    order = [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];
    idx(order) = 1:64;
    y = x(idx)';
end
```

【运行结果】运行结果如下图所示

a 8x8 double									b 64x1 double			
	1	2	3	4	5	6	7	8	1	2	3	
1	0.8244	0.8178	0.4229	0.0688	0.5313	0.1537	0.6377	0.2548	1	0.8244		
2	0.9827	0.2607	0.0942	0.3196	0.3251	0.2810	0.9577	0.2240	2	0.8178		
3	0.7302	0.5944	0.5985	0.5309	0.1056	0.4401	0.2407	0.6678	3	0.9827		
4	0.3439	0.0225	0.4709	0.6544	0.6110	0.5271	0.6761	0.8444	4	0.7302		
5	0.5841	0.4253	0.6959	0.4076	0.7788	0.4574	0.2891	0.3445	5	0.2607		
6	0.1078	0.3127	0.6999	0.8200	0.4235	0.8754	0.6718	0.7805	6	0.4229		
7	0.9063	0.1615	0.6385	0.7184	0.0908	0.5181	0.6951	0.6753	7	0.0688		
8	0.8797	0.1788	0.0336	0.9686	0.2665	0.9436	0.0680	0.0067	8	0.0942		
9									9	0.5944		
10									10	0.3439		
11									11	0.5841		
12									12	0.0225		
13									13	0.5985		
14									14	0.3196		
15									15	0.5313		
16									16	0.1537		
17									17	0.3251		
18									18	0.5309		
19									19	0.4709		
20									20	0.4253		
21									21	0.1078		

【结果分析】成功实现了 Zig-Zag 扫描。

8. 【题目描述】对测试图像分块、DCT 和量化，将量化后的系数写成矩阵的形

式，其中每一列为一个块的 DCT 系数 Zig-Zag 扫描后形成的列矢量，第一行为各个块的 DC 系数。

【主要思想】按照作业说明书的顺序，逐步对测试图像分块、DCT 和量化即可。

【核心代码】

```
c = blockproc(double(hall_gray)-128, [8 8], ...
    @(block_struct) zig_zag(round(dct2(block_struct.data)./QTAB)));
[row, col] = size(c);
res = zeros([64, row/64*col]);
for i = 1:row/64
    res(:, (i-1)*col+1:i*col) = c((i-1)*64+1:i*64, 1:col);
end
```

【运行结果】结果存储在变量 res 中。

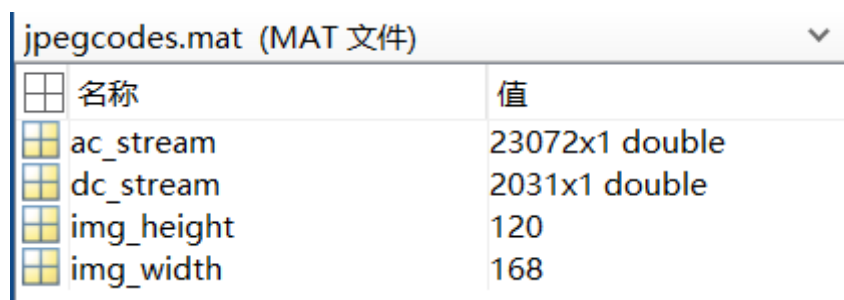
	1	2	3	4	5	6	7	8	9	10	11	12
1	57	51	9	-28	-32	-32	-32	-32	-32	-32	-32	-25
2	1	3	22	5	0	0	0	0	0	0	0	-12
3	1	-2	-40	-6	0	0	0	0	0	0	0	-5
4	1	0	0	3	0	0	0	0	0	0	0	-3
5	0	4	-2	-6	0	0	0	0	0	0	0	5
6	-1	-1	0	3	0	0	0	0	0	0	0	11
7	0	2	1	1	0	0	0	0	0	0	0	-5
8	0	-2	5	-3	0	0	0	0	0	0	0	-4
9	-1	2	-12	4	0	0	0	0	0	0	0	4
10	0	-2	0	-2	0	0	0	0	0	0	0	1
11	0	-1	0	1	0	0	0	0	0	0	0	0
12	0	1	0	-2	0	0	0	0	0	0	0	-1
13	0	-1	0	2	0	0	0	0	0	0	0	-3
14	0	1	0	-1	0	0	0	0	0	0	0	2
15	0	0	0	0	0	0	0	0	0	0	0	2
16	0	0	0	0	0	0	0	0	0	0	0	-1
17	0	-1	1	-1	0	0	0	0	0	0	0	-1

9. 【题目描述】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。

【主要思想】将上述步骤串起来，再加入 huffman 编码可以获得相应的码流。

【核心代码】

```
cD = res(1, :);
cD = [cD(1); -diff(cD)];
category = ceil(log2(abs(cD) + 1));
dc_stream = cell2mat(arrayfun(@(i) ...
    [DCTAB(category(i)+1, 2:DCTAB(category(i)+1, 1)+1),
    my_dec2bin(cD(i))]', ...
    (1:length(cD))', 'UniformOutput', false));
```

【运行结果】

A screenshot of the MATLAB variable viewer window titled 'jpegcodes.mat (MAT 文件)'. It displays a table with two columns: '名称' (Name) and '值' (Value). The variables listed are 'ac_stream' (23072x1 double), 'dc_stream' (2031x1 double), 'img_height' (120), and 'img_width' (168).

名称	值
ac_stream	23072x1 double
dc_stream	2031x1 double
img_height	120
img_width	168

10. **【题目描述】** 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。

【主要思想】 输入文件长度即为图像的像素块数量（单位是字节）乘 8，输出码流长度为 DC 和 AC 流的长度。

【核心代码】

```
CR = (img_width*img_height*8) / (length(dc_stream)+length(ac_stream));
```

【运行结果】

A screenshot of the MATLAB command window titled '命令行窗口'. It shows the output '6.4247' and the prompt 'fx >>'.

【结果分析】 压缩比为 6.4247，还是很高的。

11. **【题目描述】** 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。

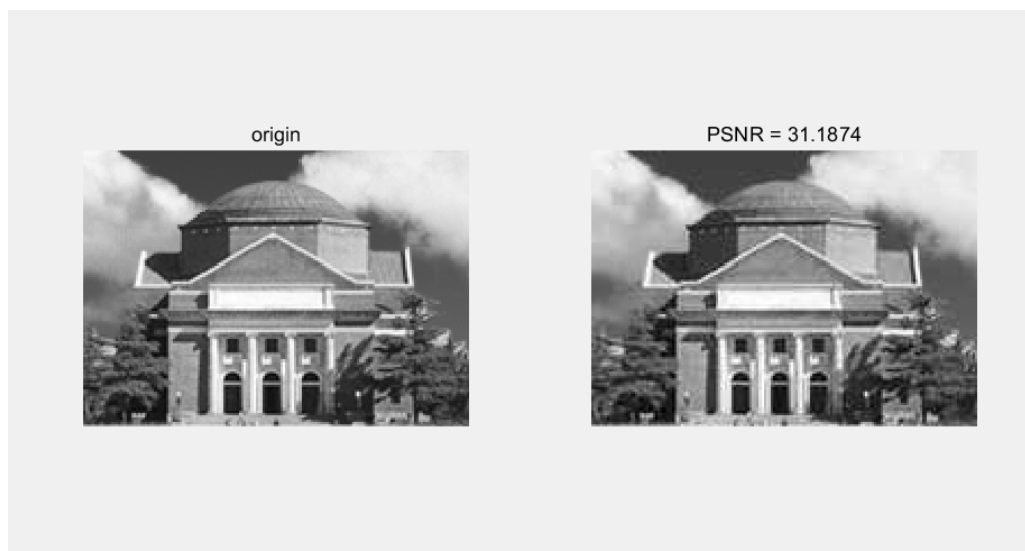
【主要思想】 解码的过程与编码过程正好相反，步骤为 DC 解码、AC 解码、重组为系数矩阵、逆 DCT 变换。DC 编码时，有三种情况，编码 00，幅度 0 序列 00；编码是三位时，幅度序列长度为十进制编码减一；编码多于三位时，幅度序列长度为第一个数加二。所以可以看 0 出现的位置判断编码。而对于 AC 编码，Huffman 码表较为复杂，故采用 ismember 和 find 结合的方式，查找序列的前 i 位是否存在于表中。对于解码得到的 64×blocks 的系数矩阵，先使用 reshape，转换为 64×横向块数×纵向块数 的张量。再使用 permute 交换行列，以满足

Matlab 列优先的特性。最后使用 reshape 和转置 变形为 64 纵向块数×横向块数的矩阵。

【核心代码】

```
while(cnt <= blocks)
    if(all(dc_stream(1:2) == [0 0]))
        dc_stream(1:2) = [];
    else
        zero_ind = find(dc_stream==0, 1);
        if(zero_ind < 4)
            huffman_code = dc_stream(1:3);
            dc_stream(1:3) = [];
            zero_ind = bin2dec(strjoin(string(huffman_code), '')) - 1;
        else
            dc_stream(1:zero_ind) = [];
            zero_ind = zero_ind + 2;
        end
        mag = bin_vec2dec(dc_stream(1:zero_ind));
        DC_coeff(cnt) = mag;
        dc_stream(1:zero_ind) = [];
    end
    cnt = cnt + 1;
end
```

【运行结果】



【结果分析】从客观上，PSNR 的值为 31 左右，足够大，说明效果还可以，并不算失真太严重。主观观察两幅图片，发现效果确实还可以。

12. 【题目描述】将量化步长减小为原来的一半，重做解编码。同标准化步长的情况比较压缩比和图像质量。

【主要思想】将 QTAB 除以 2，剩下只需要重复操作即可。

【核心代码】

```
QTAB = QTAB / 2;  
[dc_stream, ac_stream, img_height, img_width] = JPEG_encode(hall_gray,  
QTAB, DCTAB, ACTAB);  
img = JPEG_decode(dc_stream', ac_stream', img_height, img_width, QTAB,  
ACTAB);
```

【运行结果】



【结果分析】量化步长变为原来的一般使得，压缩比下降为 4.41，PSNR 变为 34.2。实际观感上，图像噪声有所改善。

13. 【题目描述】看电视时偶尔能看到美丽的雪花图像（见 snow.mat），请对其解编码。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。

【主要思想】只需要将 snow 替换掉 hall_gray 即可

【核心代码】

```
[dc_stream, ac_stream, img_height, img_width] = JPEG_encode(snow, QTAB,  
DCTAB, ACTAB);  
img = JPEG_decode(dc_stream', ac_stream', img_height, img_width, QTAB,  
ACTAB);  
  
CR = (img_width*img_height*8) / (length(dc_stream)+length(ac_stream));  
disp(CR);
```

```
MSE = sum((double(img) - double(snow)).^2, 'all') / (img_height *  
img_width);  
PSNR = 10 * log10(255 * 255 / MSE);  
disp(PSNR);
```

【运行结果】



【结果分析】相比于原来的图片，换成雪花图片之后，压缩比下降了，PSNR 也下降了。原因可能雪花图接近随机矩阵，量化表不适合对其进行处理，故最终压缩比相对较低。高频分量被量化导致高频分量存在失真，而雪花图高频分量较多，因此图片的失真比较严重，PSNR 较低。

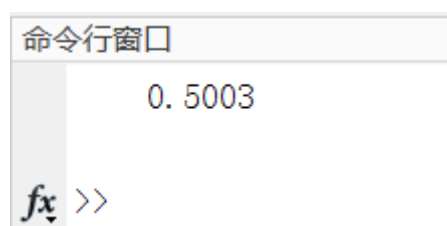
（三）信息隐藏

1. 【题目描述】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。

【主要思想】随机生成相应的加密信息，然后通过多次测试，进行相应的编码和解码，计算正确率。

【核心代码】

```
for i = 1:test_time
    seq = randi([0, 1], 1, randi(max_length));
    secret = [seq, zeros(1, max_length - length(seq))];
    encode_data = reshape(bitand(img, zeros(size(img))), 1, []) +
secret;
    encode_data = reshape(encode_data, size(img));
    [dc_stream, ac_stream, img_height, img_width] =
JPEG_encode(encode_data, QTAB, DCTAB, ACTAB);
    decode_data = JPEG_decode(dc_stream', ac_stream', img_height,
img_width, QTAB, ACTAB);
    secret_decode = reshape(decode_data, 1, []);
    secret_decode = mod(secret_decode(1:length(seq)), 2);
    correct = correct + sum(secret_decode == seq) / length(seq);
end
```

【运行结果】

【结果分析】看到正确率再 50%左右，并没有将隐藏的信息完美的保留下来。空域隐藏法抗编解码能力弱。

2. **【题目描述】**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。

【主要思想】

- (1) 用信息位逐一替换掉每个量化后的 DCT 系数的最低位，在进行熵编码。
- (2) 不是每一个系数都嵌入信息。
- (3) 先将待隐藏信息用 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 0DCT 系数之后；如果原本该图像块的最后一个系数就不为 0，那就用信息位替换该位。

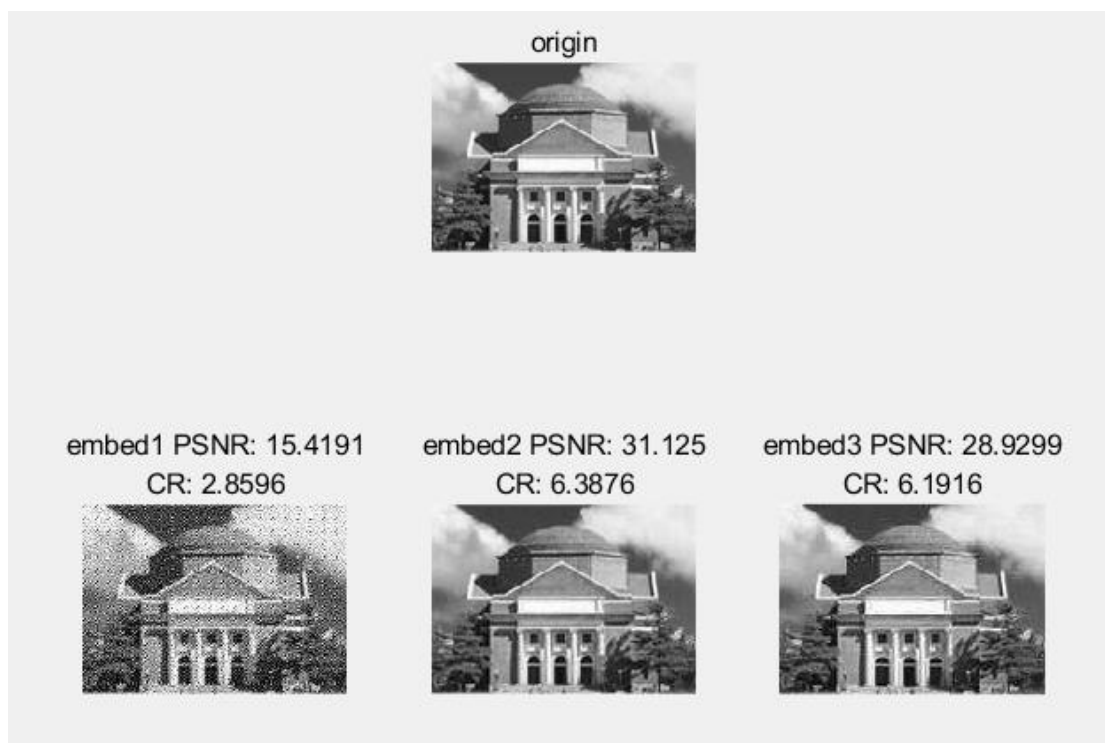
【核心代码】

```
code1 = randi([0, 1], size(c_res, 1), size(c_res, 2));
code2 = randi([0, 1], 1, size(c_res, 2));
c_hide1 = double(bitset(int64(round(c_res)), 1, code1));
```

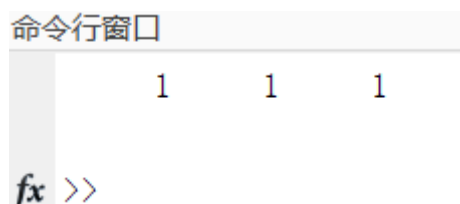


```
c_hide2 = c_res;
[~, min_idx] = min(zig_zag(QTAB));
c_hide2(min_idx, :) = double(bitset(int64(round(c_res(min_idx, :))), 1,
code2));
code3 = code2;
code3_transfrom = code3;
code3_transfrom(code3 == 0) = -1;
c_hide3 = zeros(size(c_res));
for i = 1:size(c_res, 2)
    seq = c_res(:, i);
    if seq == 0
        c_hide3(:, i) = [code3_transfrom(i); c_res(2:end, i)];
    else
        not_zero = flipud(seq ~= 0);
        seq = flipud(seq);
        [~, idx] = max(not_zero);
        if idx == 1
            seq(1) = code3_transfrom(i);
        else
            seq(idx-1) = code3_transfrom(i);
        end
        c_hide3(:, i) = flipud(seq);
    end
end
end
```

【运行结果】



正确率:



【结果分析】

方法 1 的质量和压缩比明显下降，并且看起来有明显的棋盘状图案，基本没有起到“隐藏”的意义。方法 2 和 3 添加信息较少，压缩比与原图像比时大致相同。

（四）人脸检测

1. 【题目描述】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

【主要思想】

(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？

解答：不需要，因为我们统计的是样本中各种颜色所占的比例，而与图片大小无关。

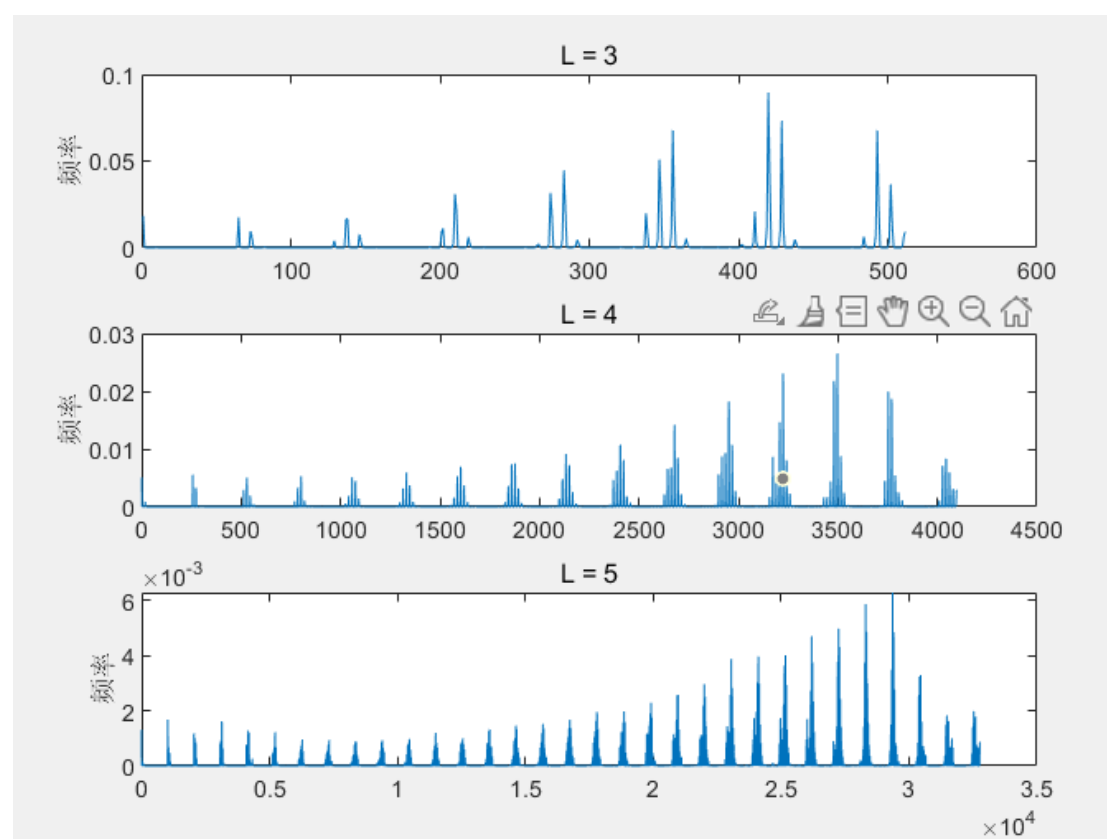
(b) 假设 L 分别取 3, 4, 5, 所得三个 v 之间有何关系?

解答: 三个 v 的关系就是相当于“分辨率”大小的关系, L 越大, 相当于将一个点的色彩范围缩得越小。 $L=3, 4, 5$ 分别对应着一个点的色彩范围位 $2^5, 2^4, 2^3$ 。

【核心代码】

```
function output = get_color_ratio(img, L)
bins = 0:2^(3*L);
img = floor(double(img) / 2^(8-L));
pixel = img(:, :, 1) * 2^(2*L) + img(:, :, 2) * 2^L + img(:, :, 3);
pixel = reshape(pixel, 1, []);
output = histcounts(pixel, bins) * 3 / numel(img);
end
```

【运行结果】



2. 【题目描述】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）。随意选取一张多人图片（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。

【主要思想】自己设置人脸的大小，然后利用框滑动的办法实现多张人脸的检测。

【核心代码】

```
for j = 0 : ww - 1
    for i = 0 : hh - 1
        block = img(i * stride(1) + 1 : i * stride(1) +
block_size(1), ...
        j * stride(2) + 1 : j * stride(2) + block_size(2), :);
        blk_hist = get_color_ratio(block, L);
        distance = 1 - sum(sqrt(blk_hist .* color_v));
        output(sub2ind([hh, ww], i + 1, j + 1), :) = ...
            [i * stride(1) + 1, j * stride(2) + 1, distance];
    end
end
```

【运行结果】



【结果分析】

L 越大，颜色细节越多，需要适当调大阈值以得到相似的结果。

3. 【题目描述】图像进行处理后，分析结果

【主要思想】分别将图像旋转 90 度，宽度拉伸 2 倍，改变颜色分析

【核心代码】

```
img_rotate = imrotate(img, 90);  
img_resize = imresize(img, [size(img, 1), size(img, 2) * 2]);  
img_adjust = imadjust(img, [.2 .3 0; .6 .7 1], []);  
  
rect_img_rotate = face_detect(img_rotate, v_3, 3, [40, 50], stride,  
0.5, 100);  
rect_img_resize = face_detect(img_resize, v_3, 3, [50, 80], stride,  
0.5, 100);  
rect_img_adjust = face_detect(img_adjust, v_3, 3, block_size, stride,  
0.5, 100);  
img_rotate = insertShape(img_rotate, 'Rectangle', rect_img_rotate,  
'Color', 'red');  
img_resize = insertShape(img_resize, 'Rectangle', rect_img_resize,  
'Color', 'red');  
img_adjust = insertShape(img_adjust, 'Rectangle', rect_img_adjust,  
'Color', 'red');
```

【运行结果】



【结果分析】

旋转没有影响检测结果；而拉伸可能会插值，使结果不同；颜色调整会带来较大的影响。这种检测方法稳定性十分不足。

4. 【题目描述】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？

【主要思想】受限于这种方法对于颜色高度敏感，人脸样本的选取应该与要检测的数据相类似，比如人的肤色需要十分相近，以及照片的光线等需要和检测数据类似，才能使这种方法达到一个比较不错的效果。

三、文件清单

本次报告的目录结构如下：

```
|— doc
|   |— report.pdf
|— res
|   |— Faces
```

```
| | | 1.bmp
| | | 10.bmp
| | | 11.bmp
| | | 12.bmp
| | | 13.bmp
| | | 14.bmp
| | | 15.bmp
| | | 16.bmp
| | | 17.bmp
| | | 18.bmp
| | | 19.bmp
| | | 2.bmp
| | | 20.bmp
| | | 21.bmp
| | | 22.bmp
| | | 23.bmp
| | | 24.bmp
| | | 25.bmp
| | | 26.bmp
| | | 27.bmp
| | | 28.bmp
| | | 29.bmp
| | | 3.bmp
| | | 30.bmp
| | | 31.bmp
| | | 32.bmp
| | | 33.bmp
| | | 4.bmp
| | | 5.bmp
| | | 6.bmp
| | | 7.bmp
| | | 8.bmp
| | | 9.bmp
| | | JpegCoeff.mat
| | | faces.jpg
| | | hall.mat
| | | snow.mat
| | src
| | | JPEG_decode.m
| | | JPEG_encode.m
| | | ex1_2.m
| | | ex2_1.m
| | | ex2_10.m
| | | ex2_11.m
```

```
| |—— ex2_12.m
| |—— ex2_13.m
| |—— ex2_2.m
| |—— ex2_3.m
| |—— ex2_4.m
| |—— ex2_5.m
| |—— ex2_7.m
| |—— ex2_8.m
| |—— ex2_9.m
| |—— ex3_1.m
| |—— ex3_2.m
| |—— ex4_1.m
| |—— ex4_2.m
| |—— ex4_3.m
| |—— face_detect.m
| |—— face_v.mat
| |—— get_color_ratio.m
| |—— jpegcodes.mat
| |—— my_dct2.m
| |—— zig_zag.m
|—— tree.txt
```

四、心得体会

这次实验让我体会到了信号与系统的有趣之处以及有用的地方。我是姚铮老师班上的学生，所以 matlab 课堂是我第一次接触谷老师，谷老师十分有趣，并且将内容活灵活现地展现在我面前。

这次图像处理作业，让我巩固了信号与系统的知识以及让我对图像处理有了初步的了解，总体来说还是非常有收获的。

但是实验过程中也有很多不足的地方：人脸识别的效果并没有做的很好；有些地方能用矩阵运算解决的地方并没有使用，而是用了较为缓慢的循环运算，等等。希望之后利用 matlab 可以避免这些问题。

在作业当中收获到很多，谢谢老师和助教的指导和帮助！

五、参考内容

- MATLAB 官方 help 文档
- 《信号与系统——MATLAB 综合实验》谷源涛 应启珩 郑君里