

第一章 基础知识

1_1

在命令窗口输入 `help images` 查看 matlab 图像处理工具箱的所有函数，阅读并大致了解这些函数的基本功能。

```
>> help images
images 的内容:

    cmap2gray - Convert colormap to grayscale colormap.
    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.
    im2gray - Convert RGB image to grayscale.
    imresize - Resize image.
    imshow - Display image in Handle Graphics figure.
    imtile - Combine multiple image frames into one rectangular tiled image.
    ind2rgb - Convert indexed image to RGB image.
    isdicom - Determine if a file is probably a DICOM file.
    isdpx - Check if file is DPX.
    isnitf - Check if file is NITF.
    rgb2gray - Convert RGB image or colormap to grayscale.
    rgb2hsv - Convert red-green-blue colors to hue-saturation-value.
    rgb2ind - Convert RGB image to indexed image.
    images - 是一个包。
    matlab - 是一个包。

    Image Processing Toolbox 文档
```

cmap2gray: 将 RGB 颜色图转换成灰度图

getrangefromclass: 根据图像类型返回默认的取值范围

hsv2rgb: 将 HSV 图像转换成 RGB 图像

im2double: 将图像（可以是灰度强度图像、真彩色图像或二值图像）转换成双精度

im2gray: 将 RGB 图像转换成灰度图像

imresize: 用于调整图像大小，将图像的长宽都缩放 `scale` 倍，默认使用双三次插值

imshow: 在图窗中显示灰度图像

imtile: 将多个图像帧合并为一个矩形分块图，返回包含 `filenames` 中指定的图像的分块图

ind2rgb: 将索引图像转换为 rgb 图像

isdicom: 检测文件是不是 DICOM 文件

isdpx: 检测文件是否包含 DPX 数据

isnitf: 检测文件是否包含 NITF 数据

rgb2gray: 消除色调和饱和度信息同时保留亮度，将 RGB 图像转换为灰度图

rgb2hsv: 将 RGB 图像的红色、绿色和蓝色值转换为 HSV 图像的色调、饱和度和明度 (HSV) 值。

rgb2ind: 将 RGB 图像转换成索引图像

1_2

(a)

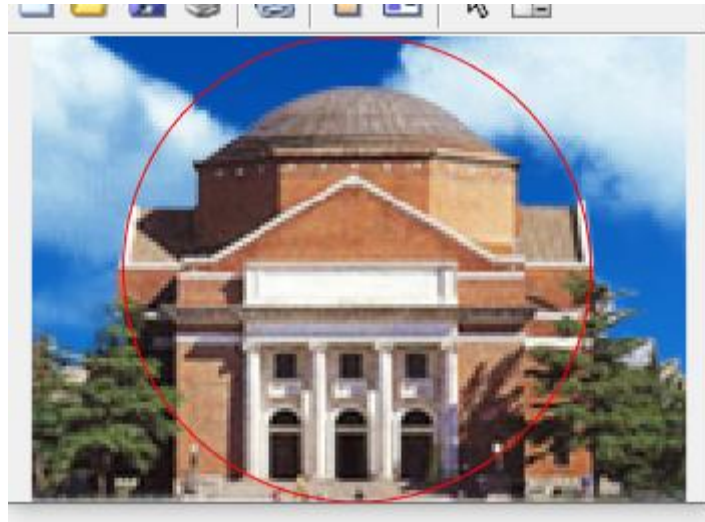
在测试图像正中央画一个内切的红色圆并用看图软件浏览

```

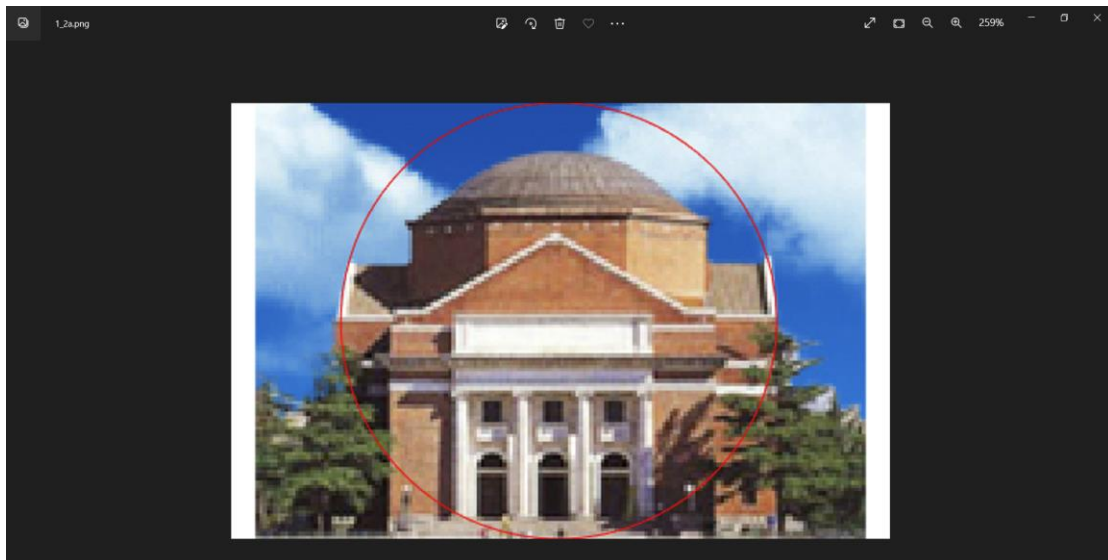
1 load('hall.mat');
2 [h,w,~]=size(hall_color);
3 r=min(h/2,w/2);
4 imshow(hall_color);
5 hold on
6 theta = 0:pi/60:2*pi;
7 x = w/2+r*cos(theta);
8 y = (h+1)/2+r*sin(theta);%直接写h/2画出来的圆有缺口
9 plot(x,y,'r');%画圆
0 set(gca,'Position',[0 0 1 1]);
1 saveas(gca,"1_2a.png");

```

思路是用 imshow 画出图后 hold on，在上面画出红色的圆，然后保存图片。其中圆的半径和圆心由读取的测试图像的高和宽决定。



matlab 显示效果



用 win10 自带的照片打开效果，可以看出图片符合要求

1_2

(b)

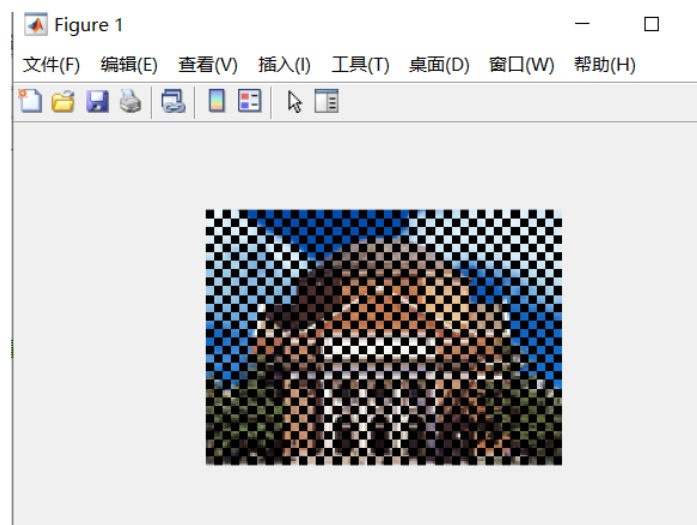
将测试图像改为黑白格形式并用看图软件浏览

```

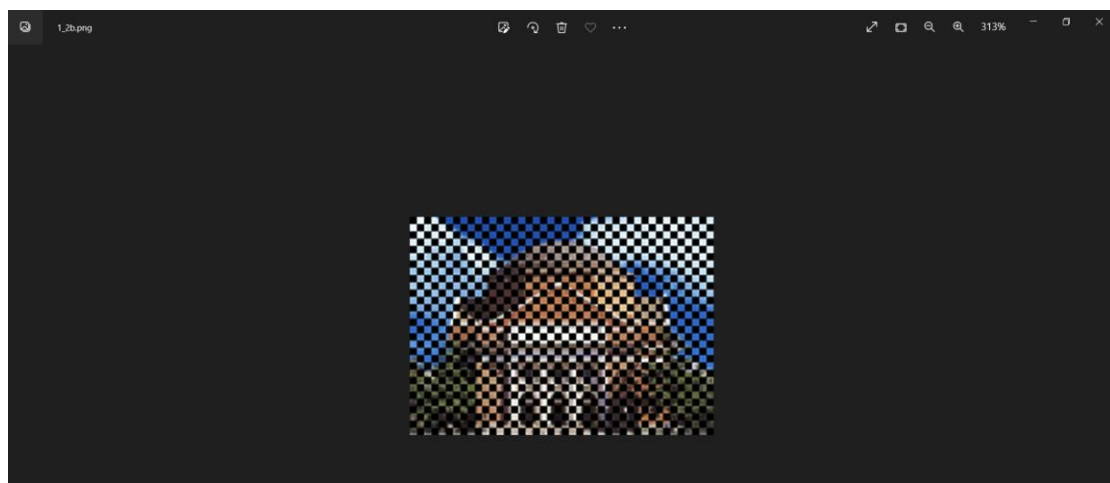
hw1_2a.m  hw1_2b.m  +
1      load('hall.mat');
2      [h,w,~]=size(hall_color);
3      n=4;%设置黑白格的边长
4      I1=(checkerboard(n,h/(2*n),w/(2*n))> 0.5);%用checkerboard函数得到黑白灰棋盘
5                                          %用>0.5的条件转化为黑白棋盘
6
7      I1=cat(3,I1,I1,I1);%I1原来只有一页，为了和全彩图运算拼接成三页
8
9      I2=hall_color;
10     I2(I1==0)=0;%彩色图对应棋盘黑色位置的数值设置为零，也就是黑色
11
12     imshow(I2);
13     imwrite(I2,"1_2b.png");

```

思路是用 checkerboard 生成黑白灰棋盘，用>0.5 的条件生成逻辑值矩阵，白色处是 1，黑色处是 0，将三个这样的矩阵拼接成三页，测试图像对应棋盘矩阵是 0 的位置置 0，即设置成黑色。



matlab 显示效果



用 win10 自带的照片打开效果，可以看出图片符合要求

第二章 图像压缩编码

2_1

图像预处理需要在灰度值上减去 128，这个操作能否在变换域执行，截取一块测试图像验证结论。

这个操作可以在变换域执行，在整体灰度值上减去 128 相当于对图像直流分量的修改，由二维 DCT 的线性性质，预处理后的图像块进行二维 DCT 变换的结果等于直接对原图像块进行二维 DCT 后减去一个 8×8 的元素全是 128 的矩阵的二维 DCT 变换结果。

```

hw1_2a.m x hw1_2b.m x hw2_1.m x +
1      load('hall.mat');
2      testpic=hall_gray(10:17,20:27);%选取测试图像中8*8的一块
3      A1=dct2(testpic);%直接对原图像中的一块进行二维DCT变换
4      preproc=dct2(128*ones(8));%得到变换域需要减去的矩阵
5      A1=A1-preproc;
6      A2=dct2(testpic-128);%预处理后进行二维DCT变换

```

A1 x A2 x									
8x8 double									
	1	2	3	4	5	6	7	8	9
1	701.5000	93.0283	-15.0323	5.5197	-0.7500	2.0683	-2.9738	1.2702	
2	47.7417	22.4016	-6.3169	6.6288	-7.2634	4.0431	-2.6570	-0.2095	
3	-29.8568	7.1690	-7.4623	8.0784	-2.9234	2.6537	-2.0481	1.2595	
4	1.0643	4.0504	0.5267	-0.8696	-2.1382	0.7947	1.0544	1.1510	
5	-7.2500	7.2015	-6.0545	2.1446	-0.5000	1.1065	-1.5511	0.4347	
6	-1.2156	5.4630	-3.6184	1.4589	0.7383	0.6595	0.5755	0.7043	
7	-4.1394	3.7277	-2.5481	-0.7186	0.1285	-0.5615	-0.0377	2.1189	
8	-1.8700	1.9548	-1.9916	1.8031	0.0318	0.5148	1.0036	-0.6914	
9									
10									
11									

A1 x A2 x									
8x8 double									
	1	2	3	4	5	6	7	8	9
1	701.5000	93.0283	-15.0323	5.5197	-0.7500	2.0683	-2.9738	1.2702	
2	47.7417	22.4016	-6.3169	6.6288	-7.2634	4.0431	-2.6570	-0.2095	
3	-29.8568	7.1690	-7.4623	8.0784	-2.9234	2.6537	-2.0481	1.2595	
4	1.0643	4.0504	0.5267	-0.8696	-2.1382	0.7947	1.0544	1.1510	
5	-7.2500	7.2015	-6.0545	2.1446	-0.5000	1.1065	-1.5511	0.4347	
6	-1.2156	5.4630	-3.6184	1.4589	0.7383	0.6595	0.5755	0.7043	
7	-4.1394	3.7277	-2.5481	-0.7186	0.1285	-0.5615	-0.0377	2.1189	
8	-1.8700	1.9548	-1.9916	1.8031	0.0318	0.5148	1.0036	-0.6914	
9									
10									
11									

可以看出得到的两个矩阵是完全一致的，也就是说预处理步骤可以在变换域进行

2_2

编程实现二维 DCT，并与 matlab 自带库函数 dct2 比较是否一致。

```

1 function [B] =mydct2(A,m,n)
2 if(~exist('m','var'))
3     m=size(A,1);%在没有传参的时候设置默认参数m
4 end
5 if(~exist('n','var'))
6     n=size(A,2);%在没有传参的时候设置默认参数n
7 end
8 if(m>size(A,1)||n>size(A,2))
9     A(m,n)=0;%用0自动扩充矩阵A到max(size(A,1),m)*max(size(A,2),n)
10 end
11 A=A(1:m,1:n);%截取A矩阵
12 D1= repmat((1:m-1)',1,m). * repmat(1:2:2*m-1,m-1,1);
13 D2= repmat((1:n-1)',1,n). * repmat(1:2:2*n-1,n-1,1);
14 D1=sqrt(2/m)*cos(D1*pi/(2*m));%得到第一行之外的算子矩阵
15 D2=sqrt(2/n)*cos(D2*pi/(2*n));
16 D1=[ones(1,m)/sqrt(m);D1];%加上第一行
17 D2=[ones(1,n)/sqrt(n);D2];
18 B=D1*A*D2';%二维DCT
19 end

```

函数首先判断是否传入 m, n 两个参数, 如果没传入则以传入矩阵的第一、二维大小为默认值; 利用 matlab 在数组下标界外赋值会自动用 0 填充的特性扩充矩阵, 截取得到需要大小的矩阵。

$$\mathbf{c} = \mathbf{D}\mathbf{p} = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \cdots & \sqrt{\frac{1}{2}} \\ \cos \frac{\pi}{2N} & \cos \frac{3\pi}{2N} & \cdots & \cos \frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(N-1)\pi}{2N} & \cos \frac{(N-1)3\pi}{2N} & \cdots & \cos \frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix}$$

思路是构造图中的算子矩阵 D, 利用矩阵乘法计算二维 DCT

矩阵构造过程用矩阵 $\begin{bmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ N-1 & N-1 & \cdots & N-1 \end{bmatrix}$ 和 $\begin{bmatrix} 1 & 3 & \cdots & 2N-1 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 3 & \cdots & 2N-1 \end{bmatrix}$ 对应元素相乘

得到 $\begin{bmatrix} 1 & 3 & \cdots & 2N-1 \\ \vdots & \vdots & \cdots & \vdots \\ N-1 & (N-1)3 & \cdots & (N-1)(2N-1) \end{bmatrix}$, 对矩阵所有元素乘 $\frac{\pi}{2N}$ 后进行 cos 运算, 再整

体乘 $\sqrt{\frac{2}{N}}$ 得到算子矩阵的第 2 到第 N 行, 第一行用 N 个 $\sqrt{\frac{1}{N}}$ 填充得到完整的算子矩阵。传入的矩阵被修改为 m×n 尺寸, 因此用 m 和 n 分别代替 N 构造算子矩阵计算即可。

$$\mathbf{C} = [\mathbf{c}_0, \mathbf{c}_1, \cdots, \mathbf{c}_{N-1}] \mathbf{D}^T \quad (2.8)$$

$$= [\mathbf{D}\mathbf{p}_0, \mathbf{D}\mathbf{p}_1, \cdots, \mathbf{D}\mathbf{p}_{N-1}] \mathbf{D}^T \quad (2.9)$$

$$= \mathbf{D} [\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_{N-1}] \mathbf{D}^T \quad (2.10)$$

$$= \mathbf{D}\mathbf{P}\mathbf{D}^T \quad (2.11)$$

利用算子矩阵的二维 DCT 计算公式

```

1 testdata=rand(8,8);%随机生成一个8*8矩阵
2 B1=mydct2(testdata);%自己编写的二维DCT
3 B2=dct2(testdata);%matlab库函数
4 B3=mydct2(testdata,10,9);%自己编写的二维DCT, 并把输入矩阵resize
5 testdata(10,9)=0;%把输入矩阵用零填充到10*9
6 B4=dct2(testdata);%用库函数进行二维DCT

```

测试用程序

B1									B2								
8x8 double									8x8 double								
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8	
1	3.8618	0.2033	0.4043	0.0523	-0.0500	-0.3265	-0.2780	0.1122	1	3.8618	0.2033	0.4043	0.0523	-0.0500	-0.3265	-0.2780	0.1122
2	-0.1853	-0.1617	-0.4013	0.0575	0.0839	-0.5190	0.2578	0.2477	2	-0.1853	-0.1617	-0.4013	0.0575	0.0839	-0.5190	0.2578	0.2477
3	0.2782	0.2049	-0.2233	0.4636	-0.0118	-0.3897	0.3853	0.2310	3	0.2782	0.2049	-0.2233	0.4636	-0.0118	-0.3897	0.3853	0.2310
4	0.2049	0.3346	-0.5683	0.1092	-0.1500	0.1652	-0.1065	0.0497	4	0.2049	0.3346	-0.5683	0.1092	-0.1500	0.1652	-0.1065	0.0497
5	-0.4783	-0.0247	0.1100	0.0174	0.0263	0.0943	0.2668	-0.0399	5	-0.4783	-0.0247	0.1100	0.0174	0.0263	0.0943	0.2668	-0.0399
6	-0.0110	0.4344	-0.0312	-0.2965	0.0498	-0.2346	0.0053	-0.1392	6	-0.0110	0.4344	-0.0312	-0.2965	0.0498	-0.2346	0.0053	-0.1392
7	0.0124	-0.1156	0.2361	-0.0014	0.0115	0.0987	-0.5432	-0.3188	7	0.0124	-0.1156	0.2361	-0.0014	0.0115	0.0987	-0.5432	-0.3188
8	0.2939	-0.2278	0.3602	-0.0159	-0.1688	0.0936	-0.0274	0.3638	8	0.2939	-0.2278	0.3602	-0.0159	-0.1688	0.0936	-0.0274	0.3638
9									9								
10									10								

B3									B4								
10x9 double									10x9 double								
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8	
1	3.2566	0.7054	-0.1927	0.6512	-0.4737	0.2621	-0.6295	0.0515	1	3.2566	0.7054	-0.1927	0.6512	-0.4737	0.2621	-0.6295	0.0515
2	0.9447	0.1186	-0.3120	0.0589	0.0129	-0.1040	-0.4432	0.2465	2	0.9447	0.1186	-0.3120	0.0589	0.0129	-0.1040	-0.4432	0.2465
3	-0.8799	-0.1968	-0.1963	-0.1276	0.5098	-0.4441	-0.0968	0.4609	3	-0.8799	-0.1968	-0.1963	-0.1276	0.5098	-0.4441	-0.0968	0.4609
4	0.9788	0.5298	-0.2615	0.2203	0.1826	-0.2465	-0.0321	0.1192	4	0.9788	0.5298	-0.2615	0.2203	0.1826	-0.2465	-0.0321	0.1192
5	-0.2701	0.1474	-0.3274	-0.2606	0.1239	-0.1368	0.2985	-0.2080	5	-0.2701	0.1474	-0.3274	-0.2606	0.1239	-0.1368	0.2985	-0.2080
6	-0.4034	-0.0952	0.1405	-0.0066	0.0523	-0.0033	0.1678	0.1846	6	-0.4034	-0.0952	0.1405	-0.0066	0.0523	-0.0033	0.1678	0.1846
7	0.1147	0.3109	0.2056	-0.1940	-0.0240	-0.0534	-0.1484	0.1700	7	0.1147	0.3109	0.2056	-0.1940	-0.0240	-0.0534	-0.1484	0.1700
8	-0.2587	0.1058	0.0531	-0.2608	-0.0153	0.0209	-0.1342	-0.4237	8	-0.2587	0.1058	0.0531	-0.2608	-0.0153	0.0209	-0.1342	-0.4237
9	0.3399	-0.1980	0.2072	0.4363	-0.2255	0.2128	-0.0823	-0.3174	9	0.3399	-0.1980	0.2072	0.4363	-0.2255	0.2128	-0.0823	-0.3174
10	0.1115	-0.1119	0.0947	0.1138	-0.2499	0.0115	0.0013	0.1310	10	0.1115	-0.1119	0.0947	0.1138	-0.2499	0.0115	0.0013	0.1310
11									11								
12									12								
13									13								

B1 和 B2 相同，B3 和 B4 相同，可以看出编写的二维 DCT 函数与 matlab 库中的 dct2 函数一致

2.3

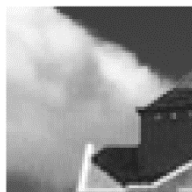
将得到的 DCT 系数矩阵右侧四列全部置 0，观察逆变换图像变化；选取一块测试图像验证结论，将系数矩阵左侧四列全部置 0 再次观察。

右侧四列系数对应高频分量，置零后图像丢失的是细节，整体和原图像相似；左侧四列对应直流分量和低频分量，由于人眼对低频分量更敏感，逆变换后得到的图像会明显变暗。

```

1 load('hall.mat');
2 testpic=double(hall_gray(1:60,1:60))-128;%选取测试图像左上角并预处理
3 B=mydct2(testpic);%二维DCT
4 B1=[B(:,1:56),zeros(60,4)];%DCT系数矩阵右侧四列置零
5 B2=[zeros(60,4),B(:,5:60)];%DCT系数矩阵左侧四列置零
6 A1=myidct2(B1)+128;
7 A2=myidct2(B2)+128;
8 figure(1)
9 imshow(uint8(A1));%DCT系数矩阵右侧四列置零后逆变换图像
10 figure(2)
11 imshow(uint8(A2));%DCT系数矩阵左侧四列置零后逆变换图像
12 figure(3)
13 imshow(uint8(testpic+128));%原图

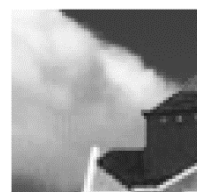
```



DCT 系数矩阵右侧四列置零



DCT 系数矩阵左侧四列置零



原图

2_4

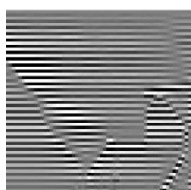
将得到的 DCT 系数矩阵转置，观察逆变换图像变化；选取一块测试图像验证结论，将系数矩阵旋转 90 度、180 度再次观察。

```
1 load('hall.mat');
2 testpic=double(hall_gray(1:60,1:60))-128;%选取测试图像左上角
3 B=dct2(testpic);
4 B_90=rot90(B);
5 B_180=rot90(B,2);
6 A1=myidct2(B')+128;%转置
7 A2=myidct2(B_90)+128;%旋转90度
8 A3=myidct2(B_180)+128;%旋转180度
9 figure(1)
10 imshow(uint8(A1));%转置后逆变换的图像
11 figure(2)
12 imshow(uint8(A2));%系数矩阵旋转90度后逆变换的图像
13 figure(3)
14 imshow(uint8(A3));%系数矩阵旋转180度后逆变换的图像
15 figure(4)
16 imshow(uint8(testpic+128));%原图
```

DCT 系数矩阵转置后逆变换得到的图像是原图像的转置， $C^T = (DPD^T)^T = (D^T)^T P^T D^T = DP^T D^T$ ，正好是原图像转置 P^T 的二维 DCT 变换。DCT 系数矩阵旋转 90 度，描述行列的系数互换位置，但交换的系数频率并不相同，逆变换得到的图像与转置类似。系数矩阵旋转 180 度，行的高低频系数互换，列的高低频系数互换，整体轮廓和原图类似。



转置



旋转 90 度



旋转 180 度



原图

2_5

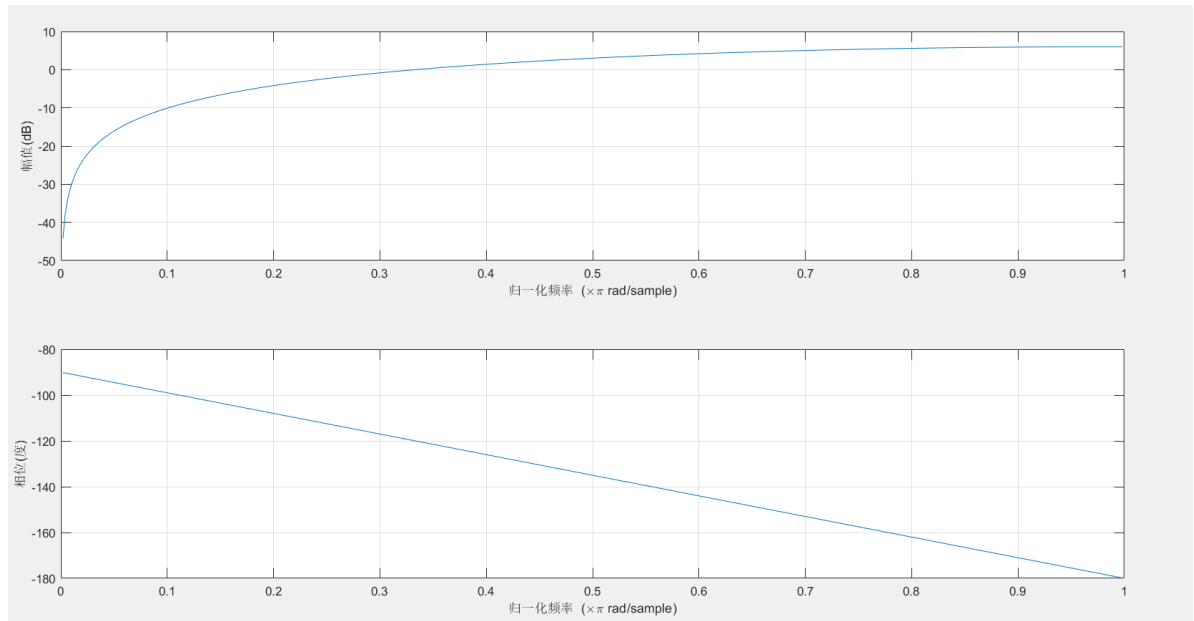
如果认为差分编码是一个系统，绘出这个系统的频率响应，说明它是一个高通滤波器，DC 系数先进行差分编码再进行熵编码，说明 DC 系数的高频频率分量更多。

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

两侧做 z 变换得到传递函数 $H(z) = z^{-1} - 1$ ，可知是高通滤波器

```
1 freqz([-1 1],1)
```

使用 freqz 画频率响应曲线



差分编码可以滤去低频分量，通过高频分量，因此 DC 系数高频分量更多，否则有用的信息都被滤去了。

2_6

DC 预测误差的取值和 Category 值有何关系？用预测误差怎么计算 Category？

预测误差是 0 时，Category 也是 0；预测误差不是 0 时，满足

$$2^{Category-1} \leq |\text{预测误差}| \leq 2^{Category} - 1$$

因此 Category = $\begin{cases} 0 & \text{预测误差} = 0 \\ \lfloor \log_2 |\text{预测误差}| \rfloor + 1 & \text{预测误差} \neq 0 \end{cases}$

2_7

列举一些实现 ZigZag 扫描的方法，借助 matlab 的强大功能设计一种最佳扫描算法

1) 打表法

将指定大小矩阵的 ZigZag 扫描下标顺序存起来，利用 Matlab 数组的下标访问可以使用另一个一维数组的特性打表实现 ZigZag 扫描输出。注意 matlab 矩阵以列为主计数。

```
1 function [B] =zigzag_1(A)
2 %打表法,只适用8*8矩阵
3 %A=reshape(1:64,8,8)';%打印下标矩阵，用来手写表格
4 Index=[1, 9, 2, 3, 10, 17, 25, 18,...
5         11, 4, 5, 12, 19, 26, 33, 41,...
6         34, 27, 20, 13, 6, 7, 14, 21,...
7         28, 35, 42, 49, 57, 50, 43, 36,...
8         29, 22, 15, 8, 16, 23, 30, 37,...
9         44, 51, 58, 59, 52, 45, 38, 31,...
10        24, 32, 39, 46, 53, 60, 61, 54,...
11        47, 40, 48, 55, 62, 63, 56, 64];
12 B=A(Index)';
13 end
14 |
```


2) 扫描法

从矩阵左上角元素开始，模仿画之字过程一个元素一个元素扫描

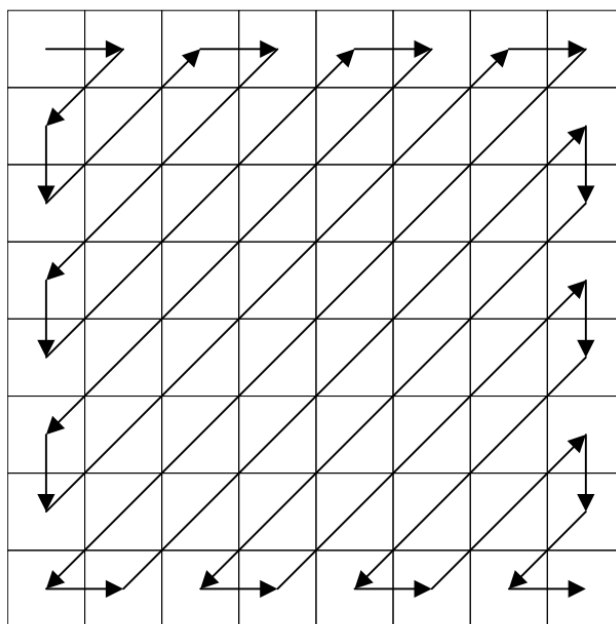


图 2.6: Zig-Zag 扫描示意图

观察扫描路径，行列下标之和为偶数时向右上扫描，遇到矩阵边界变为向右扫描，在下标变化上就是行数减 1、列数加 1，遇到边界即行数减到 1，此时只有列数加 1，在图上表现为向右扫描，行列下标和变为奇数。行列下标和为奇数时向左下扫描，每扫描一个数行数加 1、列数减 1，遇到边界，也即列数减到 1，只有行数加 1，在图上表现为向下扫描。奇偶交替直到扫描完全部的数据。

```

1 function [B] =zigzag_2(A)
2 m=size(A,1);
3 n=size(A,2);
4 B=zeros(1,n*m);
5 x=1;y=1; %从左上角开始扫描
6 k=1; %扫描后得到1×mn大小的矩阵
7 while(k<=m*n) %一共有mn个元素
8 B(k)=A(x,y); %扫描元素
9 if(mod(x+y,2)==0) %行列下标和是偶数
10 if(y==n) %碰到右侧边界，则竖直向下扫描
11 x=x+1;
12 else
13 x=max(x-1,1); %向右/右上扫描
14 y=y+1;
15 end
16 else
17 if(x==m) %碰到下侧边界，则水平向右扫描
18 y=y+1;
19 else
20 x=x+1; %向下/左下扫描
21 y=max(y-1,1);
22 end
23 end
24 k=k+1; %更新k，准备下一次扫描
25 end
26 B=B'; %转换成列向量
27 end

```

其中有较多的 if 语句，如果要实现 zigzag 扫描的矩阵数较多可以用这种方法扫描得到

方法一中的下标然后使用查表法，耗时会更短。

```
1 A=reshape(1:64,8,8)';
2 B1=zigzag_1(A);
3 B2=zigzag_2(A);
4 B3=[1, 2, 9, 17, 10, 3, 4, 11,...
5      18, 25, 33, 26, 19, 12, 5, 6,...
6      13, 20, 27, 34, 41, 49, 42, 35,...
7      28, 21, 14, 7, 8, 15, 22, 29,...
8      36, 43, 50, 57, 58, 51, 44, 37,...
9      30, 23, 16, 24, 31, 38, 45, 52,...
10     59, 60, 53, 46, 39, 32, 40, 47,...
11     54, 61, 62, 55, 48, 56, 63, 64]';
12 %B3是手动算出的矩阵Zig-Zag扫描结果
13 D1=B1-B3;
14 max(D1)
15 min(D1)
16 D2=B2-B3;
17 max(D2)
18 min(D2)
```

测试程序

```
>> hw2_7
```

```
ans =
```

```
0
```

```
ans =
```

```
0
```

```
ans =
```

```
0
```

```
ans =
```

```
0
```

可以看出 B1、B2、B3 完全相同

2_8

对测试图像分块、DCT、量化，将量化后的系数写成矩阵形式，每一列是块的 DCT 系

数进行 Zig-Zag 扫描的得到的列向量，DC 系数在第一行，也即每个块 DCT 后再扫描的结果从上到下排列。

思路先是把图像行列补成 8 的整数倍，利用 matlab 给矩阵下标范围外的位置赋值会自动扩充实现。之后把原矩阵分成每个元素是 8×8 矩阵的元胞数组，利用 cellfun 对所有块一起处理。由于没有找到类似矩阵中的 reshape 函数可以直接将元胞数组重构，写了一个二重循环把元胞数组重构，最后把元胞数组转换成矩阵返回。

```

1 function [B] =picprocess(A,QTAB)
2 [m,n]=size(A);
3 x=ceil(m/8);%纵向分块数
4 y=ceil(n/8);%横向分块数
5 if(mod(m,8)~=0||mod(n,8)~=0)
6     A(8*x,8*y)=0;%用零扩充图像
7 end
8 B=cell(1,x*y);%初始化结果元胞数组
9 C=mat2cell(double(A)-128,8*ones(1,x),8*ones(1,y));%预处理和分块
10 C=cellfun(@dct2,C,UniformOutput=false);%分块进行二维DCT变换
11 C=cellfun(@(C)QT(C,QTAB),C,UniformOutput=false);%量化
12 C=cellfun(@zigzag_1,C,UniformOutput=false);%Zig-Zag扫描
13 k=1;
14 for ii=1:x
15     for jj=1:y
16         B{1,k}=C{ii,jj};%把15x21的元胞数组改写成1x315的元胞数组，方便转换成64x315的矩阵
17         k=k+1;
18     end
19 end
20 B=cell2mat(B);%元胞数组转换成矩阵
21 end

```

图像处理函数

```

1 function [B] =QT(A,QTAB)
2 B=round(A./QTAB);
3 end

```





元胞数组不支持直接./操作，单独写了一个量化函数

```

1 load('hall.mat');
2 load('JpegCoeff.mat');
3 pic=hall_gray;
4 B=picprocess(pic,QTAB);
5

```

读取测试图像和量化矩阵，调用图像处理函数

	B	64x315 double
	DCTAB	12x10 double
	hall_color	120x168x3 uint8
	hall_gray	120x168 uint8

执行结果

64x315 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	57	51	9	-28	-32	-32	-32	-32	-32	-32	-32	-25	27	40	45	50	53
2	1	3	22	5	0	0	0	0	0	0	0	-12	-9	-4	-2	-2	-2
3	1	-2	-40	-6	0	0	0	0	0	0	0	-5	-26	-12	-5	-1	1
4	1	0	0	3	0	0	0	0	0	0	0	-3	-12	-3	0	0	-1
5	0	4	-2	-6	0	0	0	0	0	0	0	5	-5	-2	-3	-1	0
6	-1	-1	0	3	0	0	0	0	0	0	0	11	0	-1	0	0	0
7	0	2	1	1	0	0	0	0	0	0	0	-5	1	0	0	0	0
8	0	-2	5	-3	0	0	0	0	0	0	0	-4	2	-1	0	0	0
9	-1	2	-12	4	0	0	0	0	0	0	0	4	-3	-1	0	0	0
10	0	-2	0	-2	0	0	0	0	0	0	0	1	-5	-1	-1	-1	0
11	0	-1	0	1	0	0	0	0	0	0	0	0	-1	0	0	0	0
12	0	1	0	-2	0	0	0	0	0	0	0	-1	0	0	0	0	0
13	0	-1	0	2	0	0	0	0	0	0	0	-3	1	0	0	0	0
14	0	1	0	-1	0	0	0	0	0	0	0	2	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0
17	0	-1	1	-1	0	0	0	0	0	0	0	-1	0	0	0	0	0
18	0	1	1	1	0	0	0	0	0	0	0	2	-1	0	0	0	0
19	0	-1	-3	-1	0	0	0	0	0	0	0	1	-5	-1	-1	-1	0
20	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2_9

实现本章介绍的 JPEG 编码，将输出的 DC 系数码流、AC 系数码流、图像高度和图像宽度写入 jpegcodes.mat 文件

```

1 function [B] =DCcode(A,DCTAB)
2 A_temp=[2*A(1,1),A(:,1:size(A,2)-1)];
3 A=A_temp-A;%差分编码
4 B='';
5 for k=1:size(A,2)
6     if(A(1,k)==0)
7         x=1;
8     else
9         x=floor(log2(abs(A(1,k))))+2;
10    end
11    y=DCTAB(x,1);%取码长
12    for ii=2:y+1
13        B=strcat(B,int2str(DCTAB(x,ii)) );%按码长拼接对应的Huffman码
14    end
15
16    magnitudeA=dec2bin(abs(A(1,k)) );%得到二进制的magnitude
17    if(A(1,k)>=0)
18        B=strcat(B,magnitudeA);%大于0直接拼接原码
19    else
20        temp1=strrep(magnitudeA,'0','a');%利用strrep函数实现01互换，也即求1-补码
21        temp2=strrep(temp1,'1','0');
22        B=strcat(B,strrep(temp2,'a','1')) ;%拼接得到的1-补码
23    end
24 end

```

辅助函数 1，完成 DC 系数的差分编码和熵编码

```

1 function [B] =ACcode(A,ACTAB)
2 EOB='1010';
3 ZRL='11111111001';
4 B='';
5 Run=0;
6 for i=1:size(A,2)
7     A_i=A(1:size(A,1),i);
8     index=find(A_i~=0);
9     if(size(index,1)==0)%AC系数全是0
10         B=strcat(B,EOB);
11         continue
12     end
13     A_i=A_i(1:index(end),1);%去掉AC系数末尾的0
14     for j=1:size(A_i,1)
15         if(A_i(j,1)==0)
16             Run=Run+1;
17             if(Run==16)
18                 B=strcat(B,ZRL);
19                 Run=0;
20             end
21         else
22             Size=floor(log2(abs(A_i(j,1))))+1;
23             L=ACTAB(Run*10+Size,3);%取码长
24             for ii=1:L
25                 B=strcat(B,int2str(ACTAB(Run*10+Size,ii+3)));%按码长拼接对应的Huffman码
26             end
27             Amplitude_j=dec2bin(abs(A_i(j,1)));%得到二进制的amplitude
28             if(A_i(j,1)>=0)
29                 B=strcat(B,Amplitude_j);%大于0直接拼接原码
30             else
31                 temp1=strrep(Amplitude_j,'0','a');%利用strrep函数实现01互换，也即求1-补码
32                 temp2=strrep(temp1,'1','0');
33                 B=strcat(B,strrep(temp2,'a','1'));%拼接得到的1-补码
34             end
35         end
36         Run=0;
37     end
38 end
39 B=strcat(B,EOB);
40 end

```

辅助函数 2，完成 AC 系数的熵编码，为了防止末尾 0 干扰，在传入矩阵参数后，把每个块对应的列末尾的 0 全部去掉。

```

1 load('hall.mat');
2 load('JpegCoeff.mat');
3 pic=hall_gray;
4 [H,W]=size(pic);
5 pic_DCT=picprocess(pic,QTAB);%实现图像分块，二维DCT，量化和Zig-Zag扫描
6 DC_code=DCcode(pic_DCT(1,:),DCTAB);%生成DC码流
7 AC_code=ACcode(pic_DCT(2:size(pic_DCT,1),:),ACTAB);%生成AC码流
8 jpegcodes = struct('DC_code',{DC_code},'AC_code',AC_code,'H',H,'W',W);%以结构体形式存入jpegcodes.mat
9 save 'jpegcodes.mat' jpegcodes

```

主函数，将图像高度宽度和 AC、DC 码流存入 jpegcodes.mat

```

AC_code  x DC_code  x
1x23072 char

val =

'001001001110001101101010011101011110011000000110010101100101000001000001110110001000001101011010101101110000101111

```

```

AC_code  x DC_code  x
1x2054 char

1
1 11101110...

```

运行结果，DC 码流 2054bits，AC 码流 23072bits

```

1 load('jpegcodes.mat')
2 disp('图像压缩比是');
3 p=H*W*8/(size(AC_code,2)+size(DC_code,2));
4 disp(p);|

```

灰度图像大小 120*168，每个元素范围在 0-255，对应 8bit，因此在计算时乘 8 得到原图像文件大小，AC、DC 码流每一位代表 1bit，位数相加即为压缩后的图像大小

```

>> hw2_10
图像压缩比是
6.4188

```

压缩比计算结果

2_11

实现本章介绍的 JPEG 解码，以生成的 jpegcodes.mat 为输入，用客观（PSNR）方法和主观方法评价解码效果

```

1 load('jpegcodes.mat');
2 load('JpegCoeff.mat');
3 load('hall.mat');
4 B1=DCdecode(jpegcodes.DC_code,DCTAB,H*W/64);
5 AC_encodetable=[ACTAB;16,0,11,1,1,1,1,1,1,1,0,0,1,0,0,0,0,0;
6                 0,0,4,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0];%用ZRL和EOB编码填充
7 B2=ACdecode(jpegcodes.AC_code,AC_encodetable,8*8-1,H*W/64);
8 B=[B1;B2];%拼接DC和AC部分
9 pic_jpeg=pic_antiprocess(B,QTAB,H,W);
10 figure(1)
11 imshow(pic_jpeg);
12 figure(2)
13 imshow(hall_gray);
14 disp(PSNR(hall_gray,pic_jpeg));|
15

```

主要部分，调用 DCdecode 得到 DC 系数矩阵，调用 ACdecode 得到 AC 系数矩阵，拼接得到 DCT 系数矩阵，调用 pic_antiprocess 得到图片

```

>> hw2_11
31.1874

```

PSNR 结果，在正常图片的压缩范围内



解码得到的图像



原图

肉眼几乎看不出差别，因为高频分量被舍弃了，感觉解码得到的图像棱角比较平缓，比如屋檐和棚顶之间的边界就变模糊了。

```

1 function tree=buildtree(encode_table,type)
2 [m,~]=size(encode_table);
3 if(type==0)%构造DC解码的tree
4     tree(1)= struct('left',0,'right',0,'category',-1);%根节点
5     treesize=1;%树的结点数
6     for i=1:m
7         l=encode_table(i,1);%编码长度
8         index=1;%每次从根节点开始遍历
9         code=encode_table(i,2:l+1);
10        for j=1:length(code)
11            if(code(j)==0)
12                if(tree(index).left==0)%不存在左子树
13                    treesize=treesize+1;
14                    tree(treesize)=struct('left',0,'right',0,'category',-1);%构建左子树
15                    tree(index).left=treesize;%连接左子树
16                    index=treesize;
17                else
18                    index=tree(index).left;%更新index
19                end
20            elseif(code(j)==1)
21                if(tree(index).right==0)%不存在右子树
22                    treesize=treesize+1;
23                    tree(treesize)=struct('left',0,'right',0,'category',-1);%构建右子树
24                    tree(index).right=treesize;%连接右子树
25                    index=treesize;
26                else
27                    index=tree(index).right;%更新index
28                end
29            end
30        end
31    end
32    tree(index).category=i-1;%将Huffman码对应的十进制category存入节点数据域

```

辅助函数 1，用来构建解码时查找 category/run,size 的二叉树，通过 type 的值选择构建 DC 或者 AC 的查找树，区别在于 AC 树节点的数据域要存 run、size 两个数据，DC 树节点的数据域只需要存 category；AC 树在构建时有两个特殊节点，存 ZRL 和 EOB 的 run、size。

```

if(i<m-1) %针对在最后两行加入EOB和ZRL对应编码，节点数剧域填充也要分段
    tree(index).run=floor((i-1)/10);%将Huffman码对应的十进制Run、Size存入节点数据域
    tree(index).size=i-10*floor((i-1)/10);
else
    tree(index).run=16*(m-i);%对应ZRL和EOB的run
    tree(index).size=0;
end

```

对两个特殊节点特殊处理

```

1 function [B] =DCdecode(DCcodes,DCTAB,n)
2 DCTree=buildtree(DCTAB,0);
3 B=zeros(1,n);%初始化结果矩阵
4 k=1;%记录码流位置
5 for i=1:n
6     index=1;%每次循环从根节点开始
7     while(1)
8         if(DCTree(index).category~-1)%说明已经找到了一段Huffman编码对应的Category
9             break
10        end
11        if(DCcodes(k)=='0')
12            index=DCTree(index).left;
13        elseif(DCcodes(k)=='1')
14            index=DCTree(index).right;
15        end
16        k=k+1;|
17    end
18    delta_k=DCTree(index).category;%读取编码位置移动category位
19    if(delta_k==0) %category是0
20        B(i)=0;
21        delta_k=1;
22    elseif(DCcodes(k)=='1') %说明不是负数
23        B(i)=bin2dec(DCcodes(1,k:k+delta_k-1));
24    else
25        temp1=strrep(DCcodes(1,k:k+delta_k-1),'0','a');
26        temp2=strrep(temp1,'1','0');
27        temp3=strrep(temp2,'a','1');
28        B(i)=-bin2dec(temp3);
29    end
30    k=k+delta_k;
31 end
32 B(1,2:n)=-B(1,2:n);
33 for i=2:n
34     B(1,i)=B(1,i)+B(1,i-1);%逆差分编码
35 end
36 end

```

辅助函数 2，DC 系数解码，构建了查找树之后解码很简单，每次找到一个 Huffman 编码，按节点存储的 category 向后找对应位的数据，根据首位比特确定正负号，储存在结果矩阵里，最后进行逆差分运算就完成了 DC 系数解码。

```

1 function [B] =ACdecode(ACcodes,ACTAB,m,n)
2 ACTree=buildtree(ACTAB,1);
3 B=zeros(m,n);%初始化结果矩阵,m在本题中是63, n是315, 全零矩阵的好处是遇到应该填入0的位置可以直接改变j跳过
4 k=1;%记录码流位置
5 for i=1:n

```

辅助函数 3，AC 系数解码，与 DC 系数解码思路一致，细节上有所差别。AC 系数每次对其中一个块操作，因此最外层有一个 for 循环。内层有一个计数写入解码矩阵位置的 j，由于 j 变化每次不一定是加 1 因此内层有一个 while 循环。内层循环终止条件是碰到 EOB。

```

end
delta_k=ACTree(index).size;%读取编码位置移动size位
j=j+ACTree(index).run;%写入位置要下移run对应0的个数, EOB和ZRL的run也可以用这个计算
if(ACTree(index).size==0)%EOB或ZRL

```

j 变化由 run 代表要写入的 0 或者要写入的非零数据引起，第一种情况 j 自加当前节点的 run，第二种自加 1

```

1 function [B] =pic_antiprocess(A,QTAB,m,n)
2 B=cell(floor(m/8),floor(n/8));%初始化结果元胞数组
3 C=mat2cell(double(A),size(A,1),ones(1,size(A,2)));%按块把矩阵分割
4 C=cellfun(@anti_zigzag,C,UniformOutput=false);%分块进行逆Zig-Zag变换
5 C=cellfun(@(C)anti_QT(C,QTAB),C,UniformOutput=false);%反量化
6 C=cellfun(@idct2,C,UniformOutput=false);%逆DCT
7 k=1;
8 for ii=1:floor(m/8)
9     for jj=1:floor(n/8)
10         B{ii,jj}=C{1,k};%把1x315的元胞数组改写成15x21的元胞数组
11         k=k+1;
12     end
13 end
14 B=uint8(cell2mat(B)+128);
15 end

```

辅助函数 4，完成逆 Zig-Zag 变换，反量化和逆 DCT 过程

```

1 function [B] =anti_zigzag(A)
2 %打表法逆变换,只适用8*8矩阵
3 Index=[1,3,4,10,11,21,22,36,...
4         2,5,9,12,20,23,35,37,...
5         6,8,13,19,24,34,38,49,...
6         7,14,18,25,33,39,48,50,...
7         15,17,26,32,40,47,51,58,...
8         16,27,31,41,46,52,57,59,...
9         28,30,42,45,53,56,60,63,...
10        29,43,44,54,55,61,62,64];
11 B=reshape(A(Index),8,8);
12 end

```

辅助函数 5，打表法逆 Zig-Zag 变换

```

1 function [B] =anti_QT(A,QTAB)
2 B=A.*QTAB;
3 end

```

辅助函数 6，反量化，为了使用 cellfun 函数把这个部分封装成了一个单独的函数

```

1 function [PSNR] =PSNR(pic,pic_jpeg)
2 [m,n]=size(pic);
3 A=double(pic)-double(pic_jpeg);
4 MSE = sum(sum(A.^2))/(m*n);
5 PSNR = 10*log10(255^2/MSE);
6 end

```

辅助函数 7，PSNR 计算函数

2_12

将量化步长改为一半，重新编解码。和标准量化步长的情况比较压缩比和图像质量。

```

1 function [DC_code,AC_code,H,W] =JPEG(pic,DCTAB,ACTAB,QTAB)
2 [H,W]=size(pic);
3 pic_DCT=picprocess(pic,QTAB);%实现图像分块，二维DCT，量化和Zig-Zag扫描
4 DC_code=DCcode(pic_DCT(1,:),DCTAB);%生成DC码流
5 AC_code=ACcode(pic_DCT(2:size(pic_DCT,1),:),ACTAB);%生成AC码流
6 end

```

为了使用方便把之前写的编码过程包装成了一个函数，输入原图片，量化步长，DC、AC 系数的预测误差码本，输出 DC 系数码流、AC 系数码流和图像大小 H、W

```

1 function [pic_jpeg] =anti_JPEG(DC_code,AC_code,H,W,DCTAB,ACTAB,QTAB)
2 B1=DCdecode(DC_code,DCTAB,H*W/64);
3 AC_encodetable=[ACTAB;16,0,11,1,1,1,1,1,1,1,1,0,0,1,0,0,0,0;
4                 0,0,4,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0];%用ZRL和EOB编码填充
5 B2=ACdecode(AC_code,AC_encodetable,8*8-1,H*W/64);
6 B=[B1;B2];%拼接DC和AC部分
7 pic_jpeg=pic_antiprocess(B,QTAB,H,W);
8 end

```

解码的部分也包装成了函数，输入 DC 系数码流、AC 系数码流，图像大小 H、W 和两个码本，输出解码的图片。

```

1 load('JpegCoeff.mat');
2 load('hall.mat');
3 pic=hall_gray;
4 [DCcodes,ACcodes,H,W] =JPEG(pic,DCTAB,ACTAB,0.5*QTAB);
5 pic_jpeg=anti_JPEG(DCcodes,ACcodes,H,W,DCTAB,ACTAB,0.5*QTAB);
6 figure(1)
7 imshow(pic_jpeg);
8 figure(2)
9 imshow(hall_gray);
10 disp(PSNR(hall_gray,pic_jpeg));

```

主程序，调用上述编解码并把量化步长改成一半

```

>> hw2_12
34.2067

```

PSNR 结果



解码后的图像



原图

可以看出相较标准量化步长的解码结果，棱角更清晰，更大的 PSNR 也说明图像质量更好。

```

A      8x8 double
AC_code '0110011101
AC_encodet... 162x19 double
ACcod 1x34164 char
ACTAB 16x16 double

```

```

DC_code '1111011
DCcodes 1x110111 char
DCTAB 1x2423 char
H      120

```

同前，压缩比 = $\frac{8 \times 120 \times 168}{34164 + 2423} = 4.4081$ ，同标准步长的 6.4188 压缩比小了接近 1/3，但肉眼可见的图像质量提升不明显，所以一般还是使用压缩比更高的标准量化步长。

2_13

对 snow.mat 中的雪花图像进行编解码，和测试图像的压缩比和图像质量作比较并解

释原因。

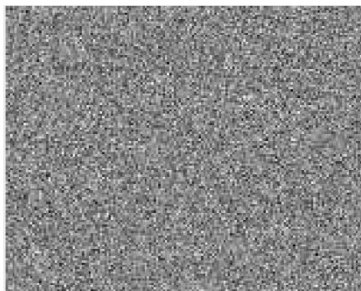
```
1 load('JpegCoeff.mat');
2 load('snow.mat');
3 [DCcodes,ACcodes,H,W] =JPEG(snow,DCTAB,ACTAB,QTAB);
4 snow_jpeg=anti_JPEG(DCcodes,ACcodes,H,W,DCTAB,ACTAB,QTAB);
5 figure(1)
6 imshow(snow_jpeg);
7 figure(2)
8 imshow(snow);
9 disp(PSNR(snow,snow_jpeg));
10 disp('压缩率: ');
11 disp(8*H*W/(length(DCcodes)+length(ACcodes)));
```

主程序，和 2_12 的主程序完全一致

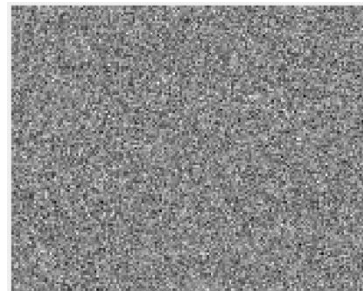
```
>> hw2_13
22.9244
```

```
压缩率:
3.6407
```

PSNR 结果和压缩率，标准量化步长的测试图像对应的 PSNR 和压缩率是 31.1874 和 6.4188。PSNR 小于 30dB，图像质量较差；同时压缩率也不高，不到测试图像压缩率的 60%



解码后图像



原图

图像压缩效果不好主要是因为雪花是随机图像，与平时见到的块内变化不算太大的图像不同，那些图像舍弃了高频成分人眼也几乎看不出，而雪花由于随机性各个频率含有信息是均等的，因此压缩效果差。

第三章 信息隐藏

3_1

实现本章介绍的空域隐藏方法和提取方法，并验证其抗 JPEG 编码能力

```

1 load('hall.mat');
2 data='matlab';
3 bindata=zeros(1,8*size(data,2)+8);% 字符对应的ascii 码用8bit即可存储
4                                     % 多出的8bit用来存储停止位，设置为8个0
5
6 for i=1:length(data)
7     bindata_i=dec2bin(data(1,i));
8     for j=8:-1:9-length(bindata_i)
9         bindata(1,8*i-8+j)=bindata_i(end+j-8)-'0';
10    end
11 end
12 %信息隐藏
13 pic=hall_gray;
14 B=reshape(pic,1,size(pic,1)*size(pic,2));%重整成1行的矩阵，方便信息隐藏
15 k=1;%标示信息已经存储到第几位
16 while(1)
17     B(1,k)=2*floor(B(1,k)/2)+bindata(1,k);%由于是最低位存信息，只要把最低位置0再加上信息位即可
18     k=k+1;
19     if(k==8*size(data,2)+9)
20         break
21     end
22 end
23 picwithdata=reshape(B,size(pic,1),size(pic,2));
24 imshow(picwithdata)
25 %信息提取
26 C=reshape(picwithdata,1,size(picwithdata,1)*size(picwithdata,2));%矩阵变形方便操作
27 data_recover='';
28 i=0;
29 while(1)
30     data_i='';
31     for j=1:8
32         data_i=strcat(data_i,mod(C(1,i+j),2)+'0');%用判断是不是偶数的方法得到信息位
33     end
34     i=i+8;
35     if(data_i=='0'+zeros(1,8))%每8位判断有没有碰到停止符
36         break
37     end
38     bin2dec(data_i);
39     data_recover=strcat(data_recover,char(bin2dec(data_i)));
40 end
41 disp('读取信息为:');
42 disp(data_recover);
43

```

隐藏信息和提取信息主要程序



隐藏信息后的图像，由于每个点的像素值与原图最多差 1，肉眼几乎看不出差别

读取信息为：

matlab

成功读取信息

```

load('JpegCoeff.mat');
[DCcodes,ACcodes,H,W]=JPEG(picwithdata,DCTAB,ACTAB,QTAB);
pic_jpeg=anti_JPEG(DCcodes,ACcodes,H,W,DCTAB,ACTAB,QTAB);
imshow(pic_jpeg)
%信息提取
C=reshape(pic_jpeg,1,size(pic_jpeg,1)*size(pic_jpeg,2));%矩阵变形方便操作
data_recover='';
i=0;
while(1)
    data_i='';
    for j=1:8
        data_i=strcat(data_i,mod(C(1,i+j),2)+'0');%用判断是不是偶数的方法得到信息位
    end
    i=i+8;
    if(data_i=='0'+zeros(1,8))%每8位判断有没有碰到停止符
        break
    end
    bin2dec(data_i);
    data_recover=strcat(data_recover,char(bin2dec(data_i)));
end
disp('读取信息为:');
disp(data_recover);

```

对隐藏信息的图像 JPEG 编解码后再读取信息



调用之前编写的 jpeg 编解码函数处理得到的隐藏信息的图像

读取信息为：

f v ½<×f0B}i °BË k-Â<-ç&0aIf0(

可以看出这种信息隐藏方式对 JPEG 编码几乎没有抵抗性

3_2

实现本章提到的三种变换域信息隐藏和提取方法，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

```
load('JpegCoeff.mat');
load('hall.mat');
pic=hall_gray;
%方法1
[DCcodes1,ACcodes1,H1,W1] =JPEGfor3_2(pic,DCTAB,ACTAB,QTAB,1);
data1=DataDecodefor3_2(DCcodes1,ACcodes1,120,168,DCTAB,ACTAB,16,1);
pic_jpeg1=anti_JPEG(DCcodes1,ACcodes1,H1,W1,DCTAB,ACTAB,QTAB);
disp('隐藏方法1 PSNR: ');
disp(PSNR(pic,pic_jpeg1));
disp('压缩率');
disp(8*H1*W1/(length(DCcodes1)+length(ACcodes1)));
```

主程序，由于三种方法在这里的代码只体现在 type 类型上，其余一模一样，因此只展示方法 1 部分。

```
1 function [DC_code,AC_code,H,W] =JPEGfor3_2(pic,DCTAB,ACTAB,QTAB,type)
2 [H,W]=size(pic);
3 pic_DCT=picprocess(pic,QTAB);%实现图像分块，二维DCT，量化和Zig-Zag扫描
4 bindata=[1,0,1,1,0,1,1,0,0,1,1,0,1,1,0,0];%只存储16位信息，为了编写方便数据选择16位数组
5 if(type==1)%替换掉每个量化后的DCT系数的最低位
6 %信息隐藏
7 pic=pic_DCT;
8 B=reshape(pic,1,size(pic,1)*size(pic,2));%重整成1行的矩阵，方便信息隐藏
9 k=1;%标示信息已经存储到第几位
10 B=2*floor(B/2);%为了改变所有的DCT系数，把DCT矩阵所有元素变成偶数，相当于所有DCT系数的信息位都变成了0，
11 %规定停止位是8个0，由于除了16个存储的信息外都是0，因此不用另设停止位
12 while(1)
13     B(1,k)=B(1,k)+bindata(1,k);%由于是最低位存信息，只要把最低位置0再加上信息位即可
14     k=k+1;
15     if(k==17)
16         break
17     end
18 end
19 pic_DCT=reshape(B,size(pic,1),size(pic,2));
20 elseif(type==2)%替换若干量化后的DCT系数的最低位
21 %信息隐藏
22 pic=pic_DCT;
23 bindata2=[bindata,zeros(1,8)];%加上停止位
24 B=reshape(pic,1,size(pic,1)*size(pic,2));%重整成1行的矩阵，方便信息隐藏
25 k=1;%标示信息已经存储到第几位
26 while(1)
27     B(1,k)=2*floor(B(1,k)/2)+bindata2(1,k);%由于是最低位存信息，只要把最低位置0再加上信息位即可
28     k=k+1;
29     if(k==17+8)
30         break
31     end
32 end
33 pic_DCT=reshape(B,size(pic,1),size(pic,2));
34 elseif(type==3)%用1，-1表示信息
35 %信息隐藏
36 B=pic_DCT;
37 bindata3=2*bindata-1;%把要存储的数据变成-1，1形式
38 for i=1:size(bindata3,2)
39     index=find(B(:,i)==0);%找到第i块最后一个非零值
40     if(size(index,1)==0)%这个图像块Zig-Zag结果全是0
41         B(1,i)=bindata3(i);%置直流系数为信息位
42     else
43         B(index(end),i)=bindata3(i);
44     end
45 end
46 pic_DCT=B;
47 end
48 DC_code=DCcode(pic_DCT(1,:),DCTAB);%生成DC码流
49 AC_code=ACcode(pic_DCT(2:size(pic_DCT,1),:),ACTAB);%生成AC码流
50 end
```

辅助函数 1，相较于之前打包的 JPEG 编码函数在量化和编码之间加入了隐藏信息部分，根据参数 type 选择隐藏类型。

```
1 function [data_recover] =DataDecodefor3_2(DC_code,AC_code,H,W,DCTAB,ACTAB,datasize,type)
2 B1=DCdecode(DC_code,DCTAB,H*W/64);
3 AC_encodetable=[ACTAB;16,0,11,1,1,1,1,1,1,1,0,0,1,0,0,0,0,0;
4 0,0,4,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0];%用ZRL和EOB编码填充
5 B2=ACdecode(AC_code,AC_encodetable,8*8-1,H*W/64);
6 pic_DCT=[B1;B2];%从DC、AC码流恢复量化后的系数矩阵
7 if(type==1||type==2)
8     %信息提取
9     C=reshape(pic_DCT,1,size(pic_DCT,1)*size(pic_DCT,2));%矩阵变形方便操作
10    data_recover='';
11    i=0;
12    while(1)
13        data_i='';
14        for j=1:8
15            data_i=strcat(data_i,mod(C(1,i+j),2)+'0');%用判断是不是偶数的方法得到信息位
16        end
17        i=i+8;
18        if(data_i=='0'+zeros(1,8))%每8位判断有没有碰到停止符
19            break
20        end
21        bin2dec(data_i);
22        data_recover=strcat(data_recover,char(bin2dec(data_i)));
23    end
24    elseif(type==3)
25        data_recover='';
26        i=0;
27        while(1)
28            data_i='';
29            for j=1:8
30                index=find(pic_DCT(:,i+j)~=0);
31                data_i=strcat(data_i,(pic_DCT(index(end),i+j)+1)/2+'0');%直接取Zig-Zag扫描后的最后非零位为信息位
32            end
33            i=i+8;
34            if(i==datasize+8)%判断读取数据停止
35                break
36            end
37            bin2dec(data_i);
38            data_recover=strcat(data_recover,char(bin2dec(data_i)));
39        end
40    end
41 end
42 end
```

辅助函数 2，从 DC、AC 码流中提取信息，同样是根据 type 选择提取类型。

```
>> hw3_2
隐藏方法1 PSNR:
    26.6635

压缩率
    6.1848




隐藏方法2 PSNR:
    31.1805

压缩率
    6.4143

隐藏方法3 PSNR:
    26.6770

压缩率
    6.4163
```

运行结果

 data1	'11'
 data2	'11'
 data3	'11'

提取出的数据，由于是随意输入的 16bit 并不是常见字符

	压缩率	图像质量	隐蔽性
方法 1	小于原图压缩率	低于 30dB, 较差	所有 DCT 系数最低位都隐藏了信息, 较差
方法 2	相比于原图压缩率略有降低, 但随着隐藏信息的增加方法 2 会趋向方法 1, 导致压缩率下降	30dB 以上, 相比原图直接 JPEG 压缩略有下降	在少量数据时隐蔽性强于方法 1, 但仍然较差
方法 3	压缩率几乎没有下降	低于 30dB, 较差	相对于方法 1、2, 方法 3 对于 Zig-Zag 扫描后的 DCT 系数处理, 相当于信息隐藏在更深一层, 隐蔽性较好

第四章 人脸检测

4_1

(a) 样本人脸大小不一, 是否需要先将图像调整为相同大小?

不需要, 这种人脸检测方法以人脸图片中提取的颜色比例作为标准, 再利用这个标准去比较, 和图像大小没有关系。

(b) 假设 L 分别取 3, 4, 5, 所得三个 \mathbf{v} 之间有什么关系?

\mathbf{v} 是长度为 2^{3*L} 的列向量, 因此 $L=5$ 对应的 \mathbf{v} 长度是 $L=4$ 对应的 \mathbf{v} 长度的 8 倍, $L=4$ 对应的 \mathbf{v} 长度是 $L=3$ 对应的 \mathbf{v} 长度的 8 倍。

4_2

设计一种从任意大小的图片中检测任意多张人脸的算法并编程实现 (输出图像在检测到的人脸上画红框), 任选一张多人照片测试程序, 并比较不同 L 值对检测结果的影响

```

1 function [v] =fetch_v(path,num,L)
2 v=zeros(2^(3*L),1,num);
3 w=zeros(2^(3*L),1);
4 for i=1:num
5     pic=imread(strcat(path,num2str(i),'.bmp'));%读取训练图片
6     pic=bitshift(double(pic),L-8);
7     pic=reshape(pic,1,size(pic,1)*size(pic,2),3);%重整成行向量, 方便按下标遍历
8     for j=1:length(pic)
9         index=bitshift(pic(1,j,1),2*L)+bitshift(pic(1,j,2),L)+pic(1,j,3)+1;%计数颜色出现次数
10        v(index,1,i)=v(index,1,i)+1;
11    end
12    v(:,1,i)=v(:,1,i)/sum(v(:,1,i));%把次数变成频率
13 end
14 for i=1:num
15     w=w+v(:,1,i);
16 end
17 v=w/num;
18 end

```

标准训练部分

```

1 function [isface] = facedetect(pic,L,d_max,l_block)
2 [v]=fetch_v('Faces/',33,L);%得到标准v
3 [H,W,~]=size(pic);
4 x=ceil(H/l_block);%纵向正方形块数
5 size_x=l_block*ones(1,x);%行方向分割方法
6 y=ceil(W/l_block);%横向正方形块数
7 size_y=l_block*ones(1,y);%列方向分割方法
8 if(20*x>H)
9     size_x=[l_block*ones(1,x-1),H+1_block-l_block*x];
10 end
11 if(20*y>W)
12     size_y=[l_block*ones(1,y-1),W+1_block-l_block*y];
13 end
14 %寻找可以判断人脸的图块
15 isface=zeros(x,y);%根据阈值判断图像块是不是满足标准，满足则对应元素置1，否则置0
16 C=mat2cell(double(pic),size_x,size_y,3);%分块
17 for ii=1:x
18     for jj=1:y
19         d=distance_uv(C(ii,jj),v,L);%得到图像块特征和标准的距离
20         if(d<d_max)%d_max作为阈值
21             isface(ii,jj)=1;
22         end
23     end
24 end
25 imshow(pic);
26 hold on
27 for i=1:x
28     for j=1:y
29         if(isface(i,j)==1)
30             if(i==x)
31                 h=size_x(end);
32             else
33                 h=l_block;
34             end
35             if(j==y)
36                 w=size_y(end);
37             else
38                 w=l_block;
39             end
40             rectangle('Position',[l_block*(j-1)+1,l_block*(i-1)+1,h,w],'LineWidth',1,'EdgeColor','r');
41         end
42     end
43 end
44 end
45 end

```

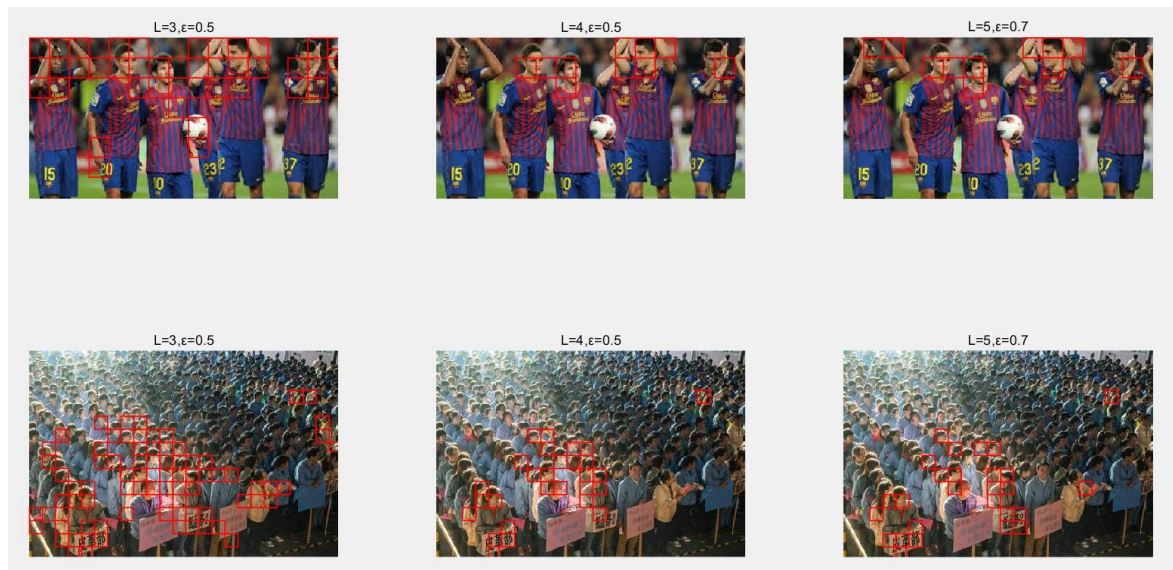
主要程序，根据输入的图片、L、图片分块数、阈值显示加了红框之后的图片。

```

1 function [d] = distance_uv(pic,v,L)
2 u=zeros(2^(3*L),1);
3 pic=bitshift(pic,L-8);
4 pic=reshape(pic,1,size(pic,1)*size(pic,2),3);%重整成行向量，方便按下标遍历
5 for i=1:length(pic)
6     index=bitshift(pic(1,i,1),2*L)+bitshift(pic(1,i,2),L)+pic(1,i,3)+1;%计数颜色出现次数
7     u(index,1)=u(index,1)+1;
8 end
9 u(:,1)=u(:,1)/sum(u(:,1));%把次数变成频率
10 d=1-sqrt(u)*sqrt(v);
11 end

```

辅助函数 1，能生成输入图像块的特征向量，并返回其与标准 v 的距离。



选取两张测试图像，L 分别选取 3，4，5 的返回结果。

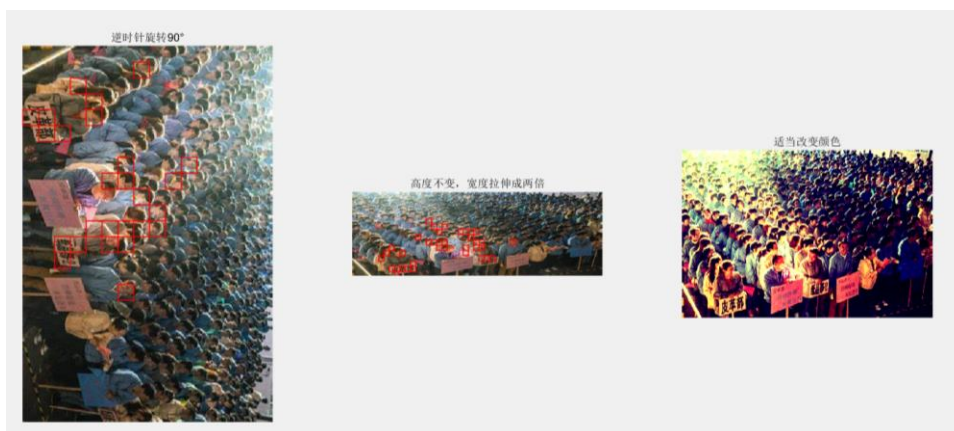
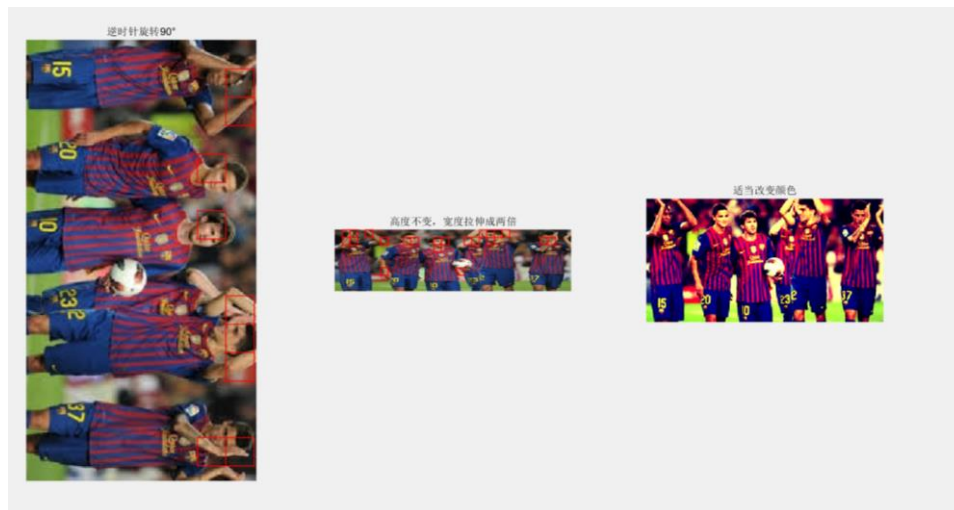
L 较小时，图像种类少，阈值设置过低会导致很多图像块误判；当 L 较大时，阈值设置也不能太高，否则会识别不出来人脸，因此三种 L 选择了不同的阈值。从测试图 1 中可

可以看出这种方法容易把胳膊上的皮肤识别成人脸，测试图 2 中把类似人脸的牌子也识别成了人脸；对人种很敏感，黑种人基本识别不出来；对光照敏感，测试图 2 中正常光照下的人容易被识别出来，站在后面阴暗处的人很难被识别正确。

4_3

对上述选择的图像做如下处理，观察并分析算法检测结果

- (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)



可以看出改变图片方向和大小对算法检测没有影响，但改变颜色对算法检测有很大影响，因为这种算法依靠颜色分布判断是不是人脸，改变图片颜色会很大程度上影响颜色的分布导致识别失败。

4_4

如果可以重新选择人脸样本训练标准，你觉得该如何选取？

我觉得首先要增大训练的图片数，训练图片以正常光照下的正脸为主，可以按比例添加一些其他肤色/光照条件下的图片，这样提取的标准应该会使检测能力提高一些，具有一些对颜色变化的鲁棒性。

文件清单：

文件名	说明
hw1_2a.m	第一章练习题2(a)代码
hw1_2b.m	第一章练习题2(b)代码
hw2_1.m	第二章练习题1代码
hw2_2.m	第二章练习题2代码
mydct2.m	自己编写的求二维DCT变换的函数
hw2_3.m	第二章练习题3代码
myidct2.m	自己编写的求二维DCT逆变换的函数
hw2_4.m	第二章练习题4代码
hw2_5.m	第二章练习题5代码,绘制频率响应
hw2_7.m	第二章练习题7代码
zigzag_1.m	查表法实现的Zig-Zag扫描函数
zigzag_2.m	自己设计的Zig-Zag扫描函数
hw2_8.m	第二章练习题8代码
picprocess.m	实现图像分块, 二维DCT, 量化和Zig-Zag扫描
QT.m	实现量化的函数, 供picprocess.m调用
hw2_9.m	第二章练习题9代码
DCcode.m	实现DC系数的差分编码和熵编码
ACcode.m	实现AC系数的熵编码
hw2_10.m	第二章练习题10代码
hw2_11.m	第二章练习题11代码
DCdecode.m	把DC码流恢复成行向量DC系数
ACdecode.m	把AC码流恢复成矩阵AC系数
buildtree.m	构建DC码流/AC码流解码时的查询树
pic_antiprocess.m	实现逆Zig-Zag, 反量化和逆二维DCT变换
anti_zigzag.m	实现逆Zig-Zag, 供pic_antiprocess.m调用
anti_QT.m	实现反量化的函数, 供pic_antiprocess.m调用
PSNR.m	计算并返回PSNR值
hw2_12.m	第二章练习题12代码

文件名	说明
JPEG.m	输入图像，输出DC、AC码流
anti_JPEG.m	输入DC、AC码流，输出图像
hw2_13.m	第二章练习题13代码
hw3_1.m	第三章练习题1代码
hw3_2.m	第三章练习题2代码
JPEGfor3_2.m	在JPEG编码的量化步骤之后可以选择三种隐藏信息方法的一种存入信息
DataDecodefor3_2.m	提取出变换域隐藏的信息
hw4_2.m	第四章练习题2代码
facedetect.m	输入需检测人脸的图像返回标红框的图像
fetch_v.m	根据训练图片提取特征 \mathbf{v}
distance_uv.m	返回输入图像块的特征和标准 \mathbf{v} 的距离
hw4_3.m	第四章练习题3代码
jpegcodes.mat	存入第二章练习题9的DC码流、AC码流、图像高H和宽W
1_2a.png	画了红色内切圆的测试图像
1_2b.png	画了棋盘格的测试图像
test1.jpg	测试图像1
test2.jpg	测试图像2