图像处理大作业

实验报告

张同和

zhang-th21@mails.tsinghua.edu.cn
2021012797
无 17
8/20/2023-8/24-2023

目录

一：基础知识

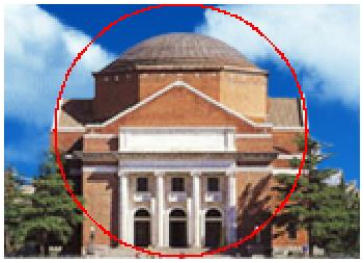
1. 熟悉 MATLAB 的图像功能(略)

2. 画圆与黑白棋盘

(a) 红色圆

以测试图像中心为圆心，图像的长和宽中较小值的一半为半径，画一个

效果展示：



关键算法说明：

由于不希望使用别的工具包，我们纯粹从数字图像的矩阵表示的视角进行操作。
考虑将原有图片的一定半径范围内的坐标点先取出来(以 bool 类常量的形式)
然后将原图这些点的 r,g,b 三个通道的色调分别调成红色全满、绿色蓝色最弱，由此则可将原图中的圆圈凸显出来。

上述操作在找出圆圈坐标点的过程中需要如下技术：

将图像中每个点的**两维坐标解耦成两个矩阵 x 与 y**，使得原图上的任意一点(x_0, y_0)具有如下性质：

$$\boldsymbol{x(x_0, y_0) = x_0}$$
$$\boldsymbol{y(x_0, y_0) = y_0}$$

这样一来，我们便可以利用 x,y 矩阵和函数表达式，将任意解析形式的图形曲线所属坐标给勾勒出来，如

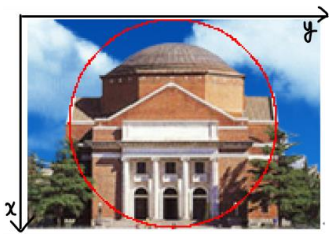
```
rim=(r-dr <=sqrt((x-cx).^2+(y-cy).^2) & (sqrt((x-cx).^2+(y-cy).^2)
<=r));
```

这就是使用圆的参数方程将原图中制定圆环内的点都抽取出来并存入名为 rim 的 bool 形矩阵的过程。这一矩阵与原图等大，其中只有圆环内的坐标为 true，其它均为 false。[1]

坐标解耦算法如下：(X,Y 分别代表图像的宽度、长度)

[1] 式中 cx,cy,r,dr 分别为图形的参数——圆心的 x,y 坐标、半径和圆环厚度。

```
shape=size(hall_gray);
X=shape(1);
Y=shape(2);
[x,y]=meshgrid(1:X,1:Y);
x=x';
y=y';
```



注意：meshgrid 函数

`[X,Y] = meshgrid(x,y)` returns 2-D grid coordinates based on the coordinates contained in vectors x and y.
X is a matrix where each row is a copy of x,
Y is a matrix where each column is a copy of y.

并不能直接完成我们的任务，因为此函数直接输出的 x,y 分别类似于:

$$x=\begin{bmatrix}1 & 2 & \cdots & X\\ 1 & 2 & \cdots & X\\ \cdots & \cdots & \cdots & \cdots\\ 1 & 2 & \cdots & X\end{bmatrix}\quad y=\begin{bmatrix}1 & 1 & \cdots & 1\\ 2 & 2 & \cdots & 2\\ \cdots & \cdots & \cdots & \cdots\\ Y & Y & \cdots & Y\end{bmatrix}$$

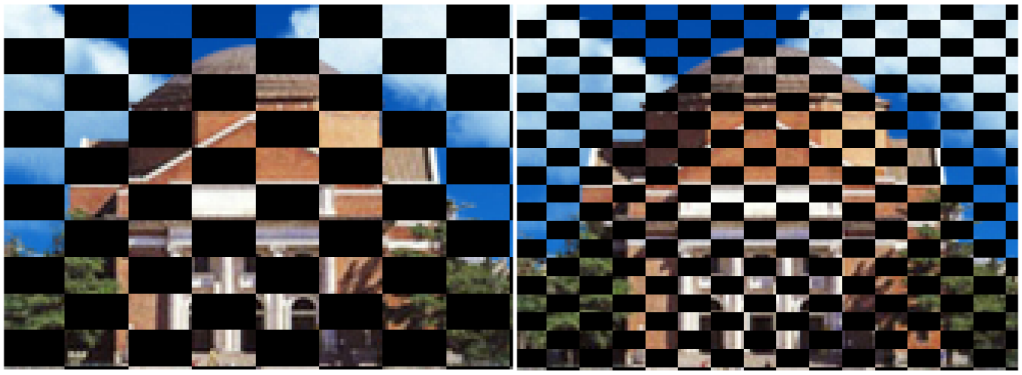
但我们实际上想要如下形式

$$x=\begin{bmatrix}1 & 1 & \cdots & 1\\ 2 & 2 & \cdots & 2\\ \cdots & \cdots & \cdots & \cdots\\ X & X & \cdots & X\end{bmatrix},\ y=\begin{bmatrix}1 & 2 & \cdots & Y\\ 1 & 2 & \cdots & Y\\ \cdots & \cdots & \cdots & \cdots\\ 1 & 2 & \cdots & Y\end{bmatrix}$$

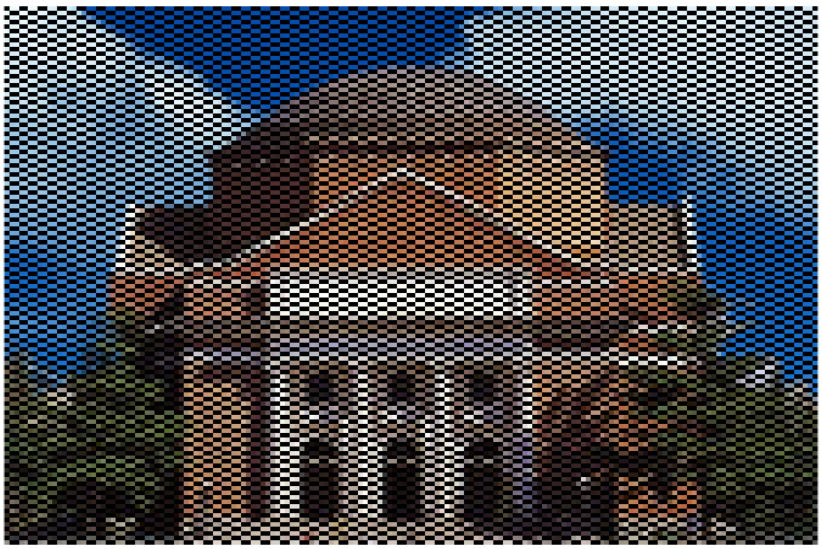
所以才有了后两部的转置。

(b)“黑白格”

效果展示：



10x8　　20 x 16



100x80[2]

核心设计

```
x_period=mod(floor(x/blk_sz(1)),2);
y_period=mod(floor(y/blk_sz(2)),2);
black_zones=(xor(x_period,y_period));
r_comp=hall_color(:,:,1);
```

由于我们希望将图像染成棋盘色，而棋盘可以被理解为将图像的等大的块状分区以 2 为周期进行交替染色。

1. 考虑到被染色的区域是**连续**出现的，黑色区域的坐标带有一定**模糊性，**黑色区域本身**又有周期性**，因此我们考虑**引入整除**：将整张图连续的坐标按照周期性地分成若干区域，实现图的分块

$$x \to floor\{\frac{x}{block_width}\}$$

然后以 2 为周期(也即“交替地”)将间隔的色块所有通道值调成 0（黑色）

2. 考虑到我们处理的是一张 **2D 图像**，黑白棋盘格要求**两个维度**上**都要**有交错变化。在 mod2 的意义下，这相当于只有对角线上的区块——横纵坐标**均属于**剩余系[0]或剩余系[1]的区块——才会被染色。因此我们直接对横纵坐标的余数求**异或**即可确定需要被染黑的区域。

		0	1	
		$(N-1)B_y+1 \cdots NB_y$	$NB_y+1 \cdots (N+1)B_y$	$\cdots$
0	$(N-1)B_x \cdots NB_x$	Black		
1	$NB_x+1 \cdots (N+1)B_x$		Black	
	$\cdots$			

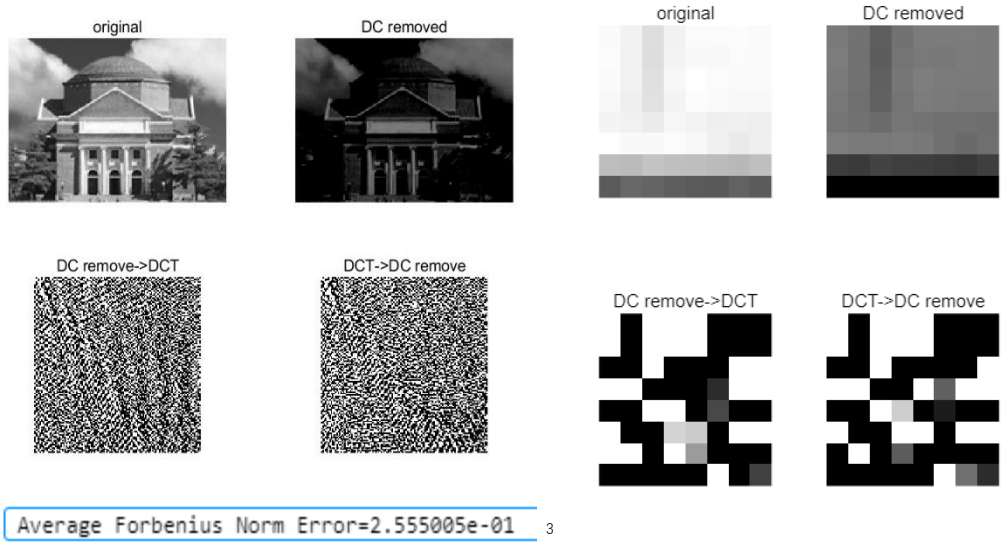
[2] 各个图下的二元数组表示这张图横、纵各被分成了多少个色块

二：图像压缩编码

1. 变换域中预处理(DC removal)

结论：可以

效果：分别截取原图中最大的正方形区域和一个长宽为 8 个小色块，用两种方法对比分析如下。其中“DC removed”指将原始照片直接去直流；“DC removal->DCT”指先在空域进行去直流，再整体做二维 DCT；“DCT->DC removal”指先整体做二维 DCT，再在频域去直流。



[3]

区域：1:120, 1:120　　区域：64:71, 56:63

原理：由于 DCT 是一种线性变换，减法与变换可交换。因此空域上先去直流再变换的结果，与先变换再在频域上减去直流的结果一致。

具体而言：

Space domain shift: $P \to \tilde{P} := P - 128 \cdot \mathbb{1}_{N\times N}$

where $\mathbb{1}$ means matrix of ones.

By the definition of DCT,

$$\tilde{C} = D\tilde{P}D^T = D(P - 128\mathbb{1})D^T = DPD^T - 128D\mathbb{1}D^T = C - 128D\mathbb{1}D^T$$

So we can see that

Frequency Domain shift: $C \to \tilde{C} := C - 128D\mathbb{1}D^T$

频域上所减去的直流矩阵$D\mathbb{1}D^T$各个元素的解析形式与其推证过程详见附录 A1.

2. 编程实现二维 DCT

并和 MATLAB 自带的库函数 dct2 比较是否一致

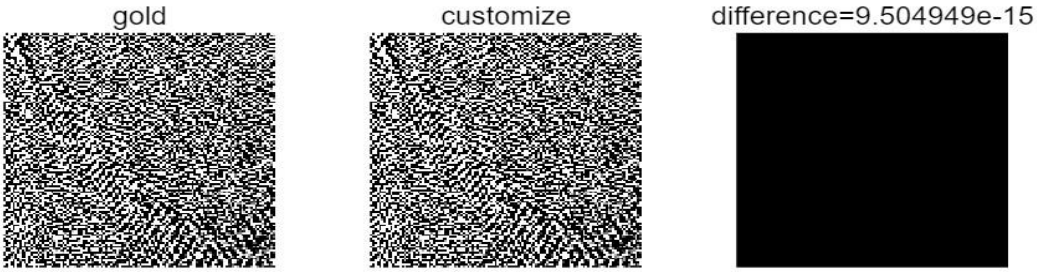
结论：高度一致。

---

[3] 这里使用两个矩阵之差的 Frobenius 范数的元素均值来度量二者的相似程度

$$\mathrm{d}(\mathrm{A},\mathrm{B}) := \frac{1}{n \cdot m}\sqrt{\sum_{i,j}(A-B)_{i,j}^2}$$

效果：下图中"gold"代表 MATLAB 内置的函数 dct2 的处理效果；"customize"代表作者实现的 DCT 函数处理效果；"difference"对应图片为两种处理之差，对应数字为两种处理结果的平均 Frobenius 范数。（见注释 2）



核心设计

```
function D=DCT_kernel(N)
a=linspace(0,N-1,N);
b=linspace(1,2*N-1,N);
raw=a'.*b;
D=sqrt(2/N).*cos(pi/(2*N).*raw);
D(1,:)=D(1,:)/sqrt(2);
end
```

DCT 变换矩阵定义如下:

Main result

By the definition of DCT operator,

$$\mathbf{D} = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \cdots & \sqrt{\frac{1}{2}} \\ \cos\frac{\pi}{2N} & \cos\frac{3\pi}{2N} & \cdots & \cos\frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos\frac{(N-1)\pi}{2N} & \cos\frac{(N-1)3\pi}{2N} & \cdots & \cos\frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix} \quad D^{-1} = D^T$$

$$D_{i,j} := \sqrt{\frac{2}{N}} \begin{cases} \sqrt{\frac{1}{2}} & ,i = 1 \\ cos(\frac{(i-1)\pi}{2N} \cdot (2j-1)) & ,2 \le i \le N \end{cases}$$

$$= \begin{cases} \sqrt{\frac{1}{N}} & ,i = 1 \\ \sqrt{\frac{2}{N}} cos(\frac{(i-1)\pi}{N} j - \frac{i-1}{2N}\pi) & ,2 \le i \le N \end{cases}$$

为了用矩阵语言来定义它，我们考虑如下事实:

1. (对矩阵逐点作用的算子): $\left(f\left(A_{i,j}\right)\right)_{(i,j)\in[n]\times[m]} = f(A)$.

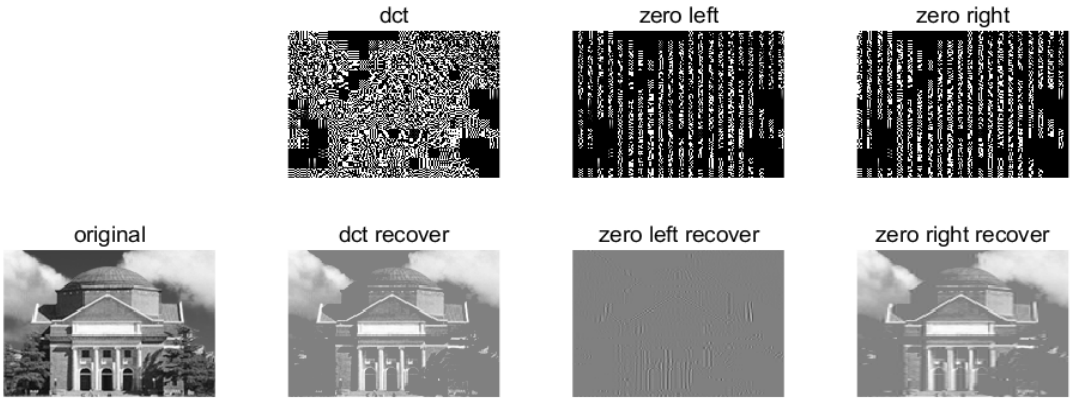
   这使得我们可以先将 D 矩阵中各个元素均具有的结果（cos，$\sqrt{\frac{2}{N}}$等）都提出来，然后观察各个元素之间的不同之处。

2. (张量积): $\left(A_{i,j}\right)_{(i,j)\in[n]\times[m]} = \left(f_i \cdot g_j\right)_{(i,j)\in[n]\times[m]} \Rightarrow A = \vec{f} \otimes \vec{g}$

   这利于我们将二维的两个方向上的变化拆成两个向量各自一个方向上的变化，从而**避免使用二重循环定义矩阵**。

另外，由于 D 矩阵的第一行较为特殊，我们对所有元素定义完之后再回去修正一步。

3. DCT 矩阵置零影响复原



注解:

这七张图中，"original"代表原图片；"dct"与"dct recover"分别代表原图的二维 DCT 结果和由此结果进行逆变换后的复原图；"zero left"与"zero left recover" 分别代表将原图的 DCT 逐块置零左四列之后的参数矩阵和据此进行逆变换的复原图；"zero right"与"zero right recover"与前述两张图定义类似，不过所置零的列为每个块的右四列。

解释:

我们可以清晰地发现如果将左半侧置零则将彻底毁灭图像的主要信息。

这主要是因为人眼观测多数图像时以关注低频信号为主，而 DCT 谱的左侧各列对应低频基底的幅度，如果抹去低频分量则会导致复原图像丧失大量人眼可辨的信息，导致图像严重模糊。

与之相反的是，若将右半侧置零，则只会消除高频信号，人眼一般并不敏感，因此第三组操作的复原结果与直接进行正逆变换相仿。

关键设计:

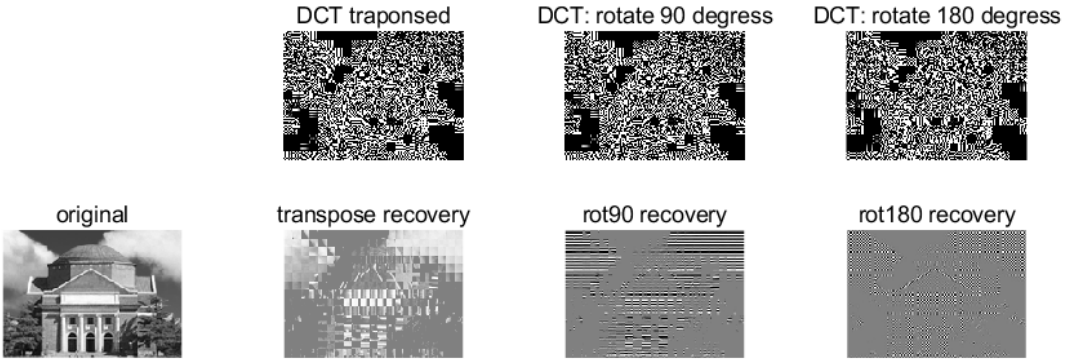
1. 由于我们希望对原图按照 8x8 的块逐个逐个地进行正逆余弦变换，我们需要使用 image processing toolbox 中的 blockproc 函数。
2. 三种不同处理的正变换彼此不同，但是逆变换是一致的。这三组操作之前都需要先去直流、操作后均**需恢复直流。**
3. 还要注意数据类型：DCT 变换之前，最好将矩阵转换为浮点形式；**IDCT 变换后要转回 uint8 格式——否则 imshow 将无法执行。**
   img→uint8 →DCT→double→IDCT→uint8→img

4. DCT 矩阵旋转影响复原

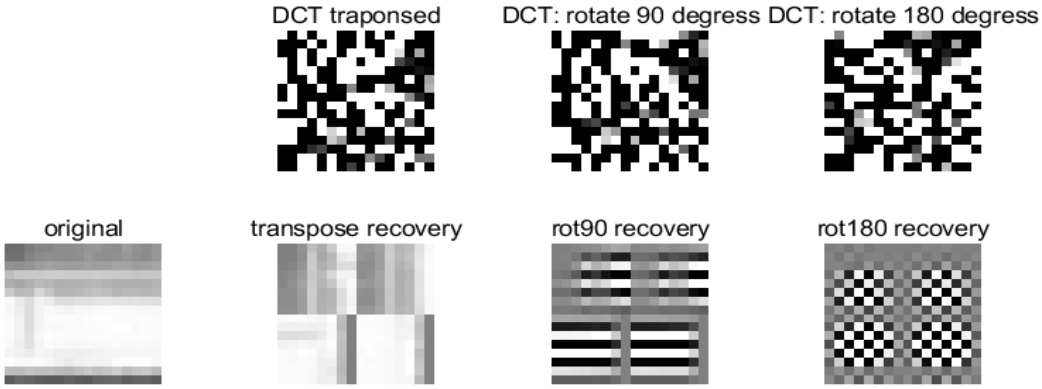
分别做转置、旋转 90 度和旋转 180 度操作（rot90)，逆变换后恢复的图像有何变化？选一块图验证你的结论。

结论: 整体上看复原的图像都严重失真。复原后失真程度：旋转 180>旋转 90>转置

效果:



所选区域：全部



所选区域：56:71,56:71

分析：严重失真主要有两个因素

a) 操作时分块进行的，会破坏整体的语义信息

b) 只在正变换时叠加了上述空间变换，而逆变换时并未补偿响应逆变换，因此必然改变了每个块的信息。

5. 差分编码

绘出差分编码系统的频率响应

它是一个-----（低通、高通、带通、带阻）滤波器。

DC 系数先进行差分编码再进行熵编码，说明 DC 系数的----频率分量更多。

2.16 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 ？

a) 差分编码定义如下：

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

忽略 n=1 初值，则其频率响应为：

$$y(n)=-x(n)+x(n-1)$$

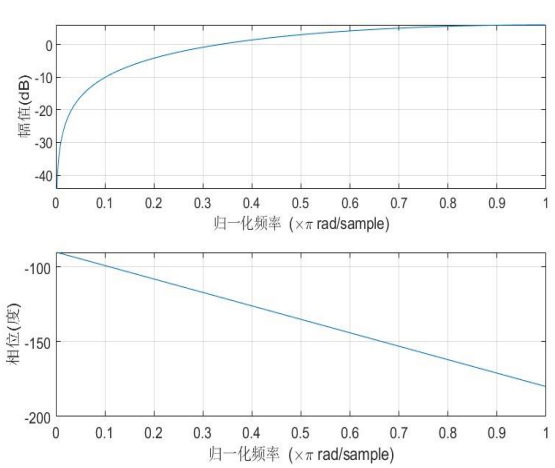
$$Y(z)=-X(z)+z^{-1}X(z)=(-1+\frac{1}{z})X(z)$$

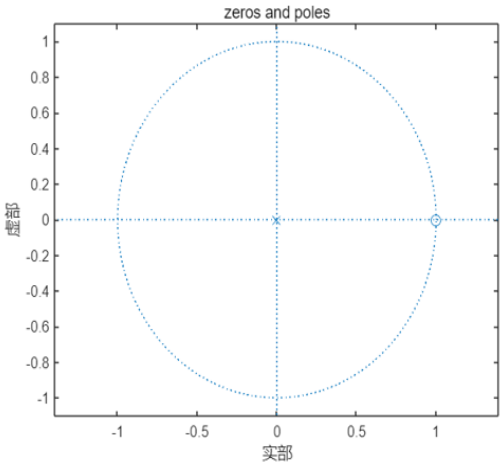
$$H(z)=\frac{Y}{X}(z)=-1+\frac{1}{z}=\frac{-1+1\cdot z^{-1}}{1}$$

b)

参考 freqz docs 、plot_zeros_poles、 difference_eqs_to_transfer_funct 两个文档

用 freqz 对此滤波器的**幅频响应、相频响应和零点极点分布图**作图如下：





说明这一系统是一个**高通滤波器。**

c) 由于先过了此差分编码再做熵编码，而差分编码倾向于滤过高频分量，因此 **DC 系统的高频分量更多**。

6. DC 预测误差的取值与 Category 的关系

如何利用预测误差计算出其 Category?
解答: 根据指导书 14 页定义

预测误差	Category
0	0
-1, 1	1
-3, -2, 2, 3	2
-7, ⋯, -4, 4, ⋯, 7	3
-15, ⋯, -8, 8, ⋯, 15	4
-31, ⋯, -16, 16, ⋯, 31	5
-63, ⋯, -32, 32, ⋯, 63	6
-127 ,⋯, -64, 64, ⋯, 127	7
-255, ⋯, -128, 128, ⋯, 255	8
-511, ⋯, -256, 256, ⋯, 511	9
-1023, ⋯, -512, 512, ⋯, 1023	10
-2047, ⋯, -1024, 1024, ⋯, 2047	11

若记预测误差为E，类别为C,则**由预测误差向类别的映射为**

$$C = \lceil log_2(|E| + 1) \rceil \quad |E| < 2048$$

7. 实现 Zig-Zag 扫描

效果：

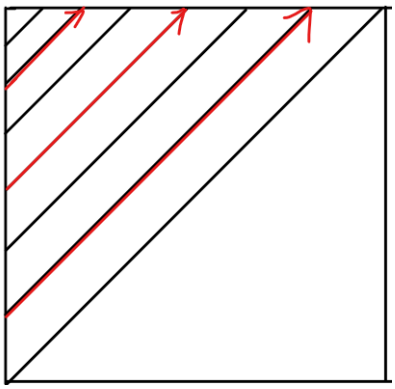
```
a =

    0.1839    0.9027    0.3377    0.7803
    0.2400    0.9448    0.9001    0.3897
    0.4173    0.4909    0.3692    0.2417
    0.0497    0.4893    0.1112    0.4039
```

```
ans =

 列 1 至 12

   0.1839    0.9027    0.2400    0.4173    0.9448    0.3377    0.7803    0.9001    0.4909    0.0497    0.4893    0.3692

 列 13 至 16

   0.3897    0.2417    0.1112    0.4039
```

算法设计

zigzag 可以被理解为两个操作的结果:

1. 将整个矩形区域内的每条斜率为 1 的斜线上的点都找出来，串在一起
2. 以 2 位周期交替地将加入进来的斜线上反过来(一上一下、一上一下)



斜线族可表示为:

$$\bigcup_{k\in\{2,\cdots,W+H\}}\{i+j=k|i,j\in\mathbb{Z}_+\}$$

其中(i,j)为斜线上点的坐标，W,H 为矩形区域的宽、高

更精细的分析能知道每条线上的 i 的取值范围实际上是

$$\{max(1,k-H),\cdots,min(k-1,W)\}$$

由此得到如下简单代码:

```
%zigzag
function z=zigzag(A)
sz=size(A);
W=sz(1);
H=sz(2);
N=W+H

%each right slanting line: i+j=k
slant=[];
%output
z=[];
```

```
%whether should flip this line
will_flip=0;
for k=2:N
    slant=[];
    will_flip=1-mod(k,2);
    for i=max(1,k-H):min(k-1,W)
        slant=[slant,A(i,k-i)];
    end
    if will_flip
        slant=flip(slant);
    end
    z=[z,slant];
end
end
```

8. 对测试图像分块、DCT 和量化，量化系数写成矩阵形式

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

设计细节： $(\boldsymbol{H},\boldsymbol{W}) \rightarrow \left(\mathbf{64},\frac{\boldsymbol{W}\cdot\boldsymbol{H}}{\mathbf{64}}\right)$　　$(\mathbf{120},\mathbf{168}) \rightarrow (\mathbf{64},\mathbf{315})$

1. 将全图切割为 8x8 的块
2. 对每个块：
   DCT 操作
   点除 Q 值
   zigzag
   [注意，由于要使用 blockproc 函数，blockproc 返回的应该是 H/8 x W/8 个块，每个块实际上是一个 64x1 的列向量，每个块的最顶格元素是这个块的$C_{0,0}$ DC 系数
   blockproc 生成的矩阵的尺寸是

$$(\frac{H}{8}\times 64,\quad \frac{W}{8})$$

3. 将 blockproc 输出结果整理为 $(64,\ \frac{H\cdot W}{64})$ 的形式

   为此需要以每 64 行为一个大区(共$\frac{H}{8}$个)，宽度为$\frac{W}{8}$

   将 blockproc 各个大区整区整区地接在一起

9. 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.nat 文件。

DC 编码：

1. 8 问的第一行已得到 DC 系数整数表示$\widetilde{c_D}$ shape=$\left(\frac{H\cdot W}{64}\right)$x1，每元素十进制整数
2. 然后需要进行**<u>差分编码</u>**生成预测误差$\widehat{c_D}$向量 shape=$\left(\frac{H\cdot W}{64}\right)$x1 元素都是十进制整数

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

3. 将预测误差向量各个元素映到 Category 上(见这一问)
4. 再根据 Category 查“DTAB”表进行熵编码，
   每个十进制数会得到一个长为 3 到 10 不等的二进制 Huffman 码

DTAB	是一个 12x10 的正整数表格 每行对应一个 category：　category [0]~[11]----row [1]~[12] 每行的第一个元素表示此类别的 Huffman 码在零延拓之前的真实长度 每行的 2~10 为类元素的 Huffman 码向右零延拓的结果 9'b…

再将每个元素的 Huffman 码与其 1 补码拼接在一起，形成一个元素的表示
并将所有元素的二进制表示都横着拼在一起，就得到了该图像的 DC 系数码流
<u>注意：直流编码时本应将 0 编码为 001，其中 00 为其 Huffman 码, 1 为其补码。但是考虑到只有 0 会对应到 00 的 huffman 码，因此可以省略 0 的补码，因而将 0 编码为</u>

00.这样可以减少许多编码空间

AC 编码:

将整个图片的 blockproc 输出结果 $(64, \frac{H \cdot W}{64})$ 中的第二行及以后的部分

每一列：量化，然后送入熵编码

ATAB 是一个 160x19 的正整数表格 1 列：Run 2 列：Size 3 列：AC-Huffman 码长 4~19 列：Huffman 码(长度不够则向右零延拓) *注意没有包含 Run,Size==00 的情况 *(Run,Size)=(0,1)~(15,10) *注意也没有包含 Run,Size==15,0 的情况

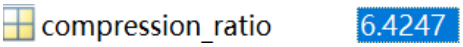
参考资料：

find nonzero elements

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。

结果：压缩比为 6.43



设计：

```
%calculate compression ratio
%note that input is an uint8 array,
%while output is binary. To uniform the format
%we see each digit of uint8 input as an 8-digit binary input
compr_r=double(h*w*8)/(double(dc_len+ac_len));
fprintf("compression rate=%.2f",compr_r);
```

11. 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。

结果:

进行评价使用的脚本请见附录 B 1

客观评价: 

31.1874

峰值信噪比较高，恢复程度较好。

主观评价:

主观来看，恢复结果较原图略有模糊，但整体上令人满意。

设计细节:

技术细节太过复杂，无法用文字简单概括，细节详见下列代码文件。

这些文件中的代码均有详细易读的注释，每两三步就有注解。另外，代码也对许多文件的格式等做了直观的说明。

编码所用文件	解码所用文件
Encode.mlx	Decode.mlx
DC.mlx	DC_decode.mlx
dc_difference_encoding.mlx	
dc_entropy_encoding.mlx	
AC.mlx	AC decode.mlx
ac entropy encode.mlx	
to_category.mlx	
complerment_1.mlx	icomplement_1 .mlx
img_to_q.mlx	q_to img.mlx
zigzag.m	izigzag.m
JpegCoeff.mat	JpegCoeff.mat
jpegcodes.mat	jpegcodes.mat
hall.mat	recover.mat

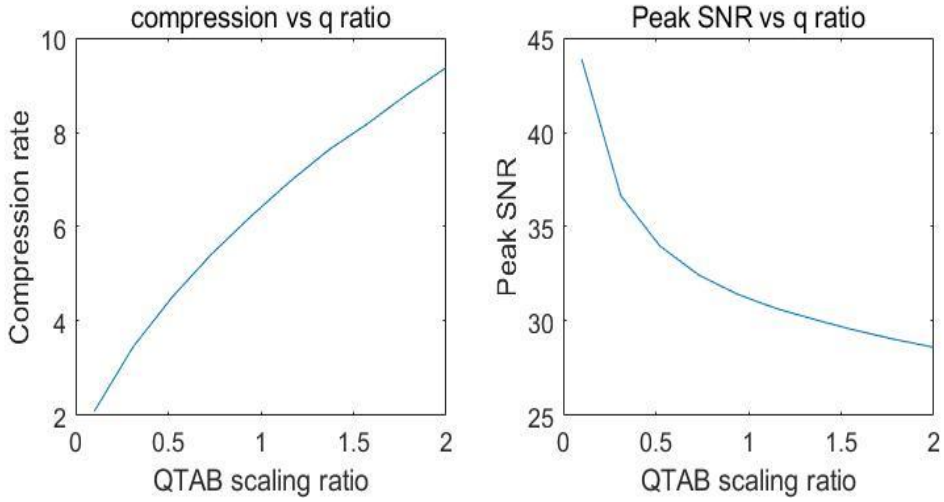
12. 量化步长减小为原来的一半，重做编解码。

同标准量化步长的情况比较压缩比和图像质量

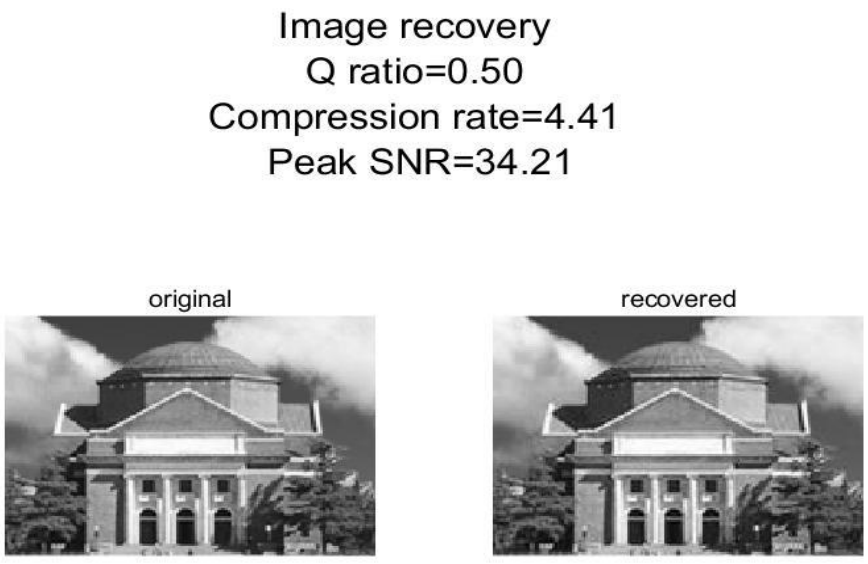
定量结果：

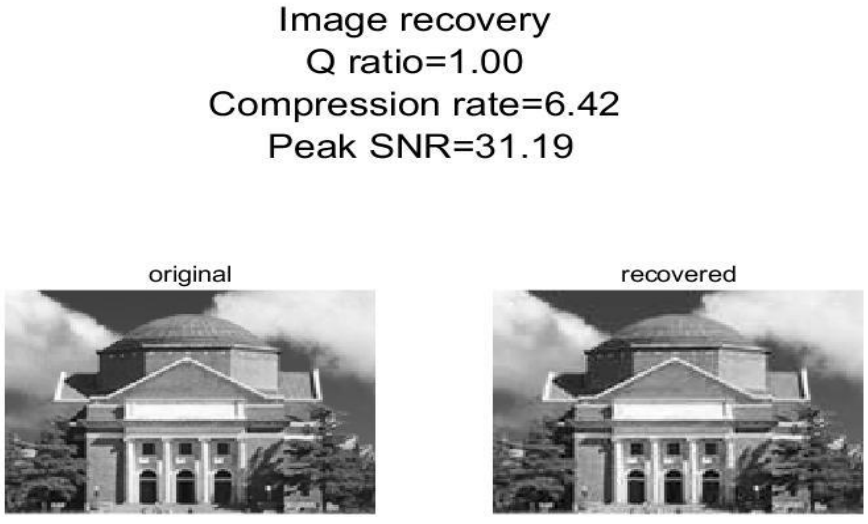
我们用 QTAB scaling ratio 来连续地模拟量化步长的整体改变，可以明显地看出

量化步长越大，压缩率越高，但是失真度越大。这与指导书上 13 页的结论一致。



定性观测不同量化步长下的图片也能得到相同的结论:





设计细节：代码详见 control_q_ratio.mlx,所做的改动只是进行了一个封装。

13. 雪花图像编-解码

电视时偶尔能石到美丽的雪花图像（见 snow .mat)。请对其编解码。和测试图像的压缩比和图像质量进行比较，并解释比较结果。

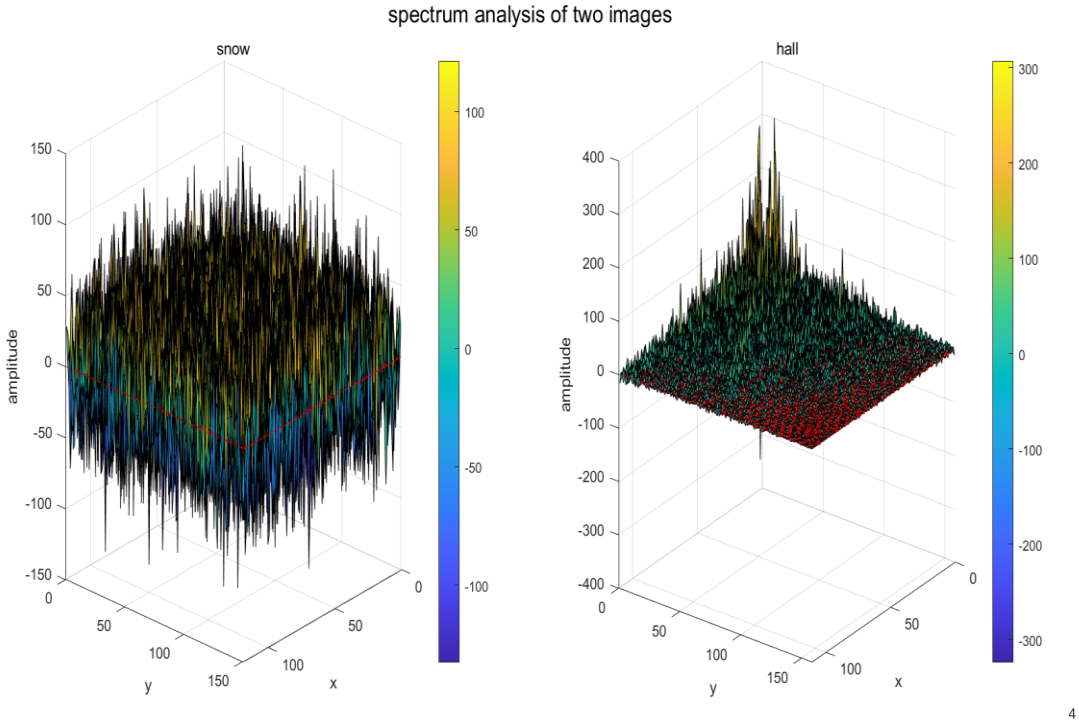
结果：

分析：

雪花图样的 PeakSNR 明显低于大礼堂，这说明我们使用的 JPEG 压缩编码技术对这类图像的恢复性能低于大礼堂。这可能是因为我们所用方法削弱了许多高频分量，而大礼堂的高频分量较小，影响少；雪花的高频分量多，影响较为严重。

如果我们对两个图像的频谱作图，并以红色平面表示全部频谱上的强度均值，可以发现 hall 的高频区明显低于均值，而 snow 的则与均值没有明显区别。因此，雪花图像的高频区域频率组分与低频相仿，占比较大；而礼堂图片的高频分量明显小于低频分量，因此我们的 JPEG 压缩算法的恢复性能更佳。

作图使用的代码在 freq_analysis.mlx 中。

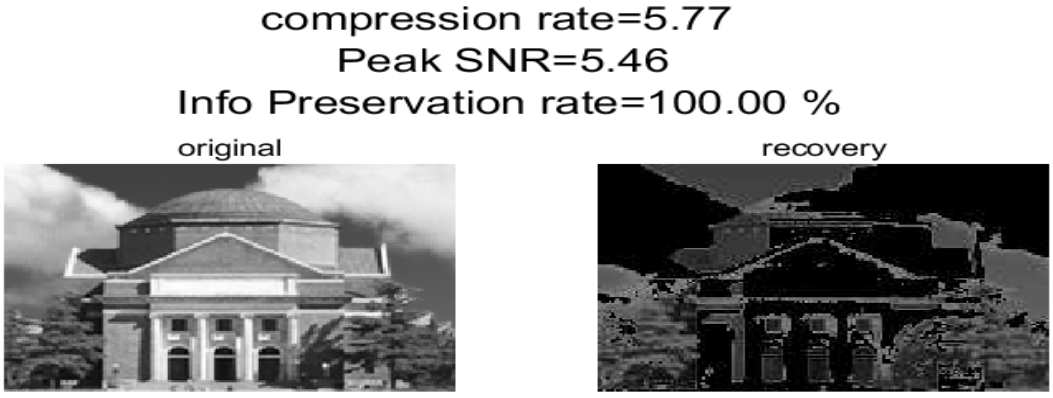


设计细节:
代码详见 EnDeCoder.mlx，主要改动是增加了可视化和改变了输入数据。

三：信息隐藏(steganography.)

1. 空域方法

将信息表示为二进制码流，再将每位信息替换掉像素亮度分量最低位。



[4] 注意：作图时舍弃了最低频率的分量区域(0,10),(0,10)，因为两张图均在最低频区域有非常强的冲激，会导致其他区域的拓扑结构变化被衬托得并不明显。

设计细节：参照指导书

```
if (type==0)
    raw_stg=dec2bin(raw);
    raw_stg(:,end)=dec2bin(info);
    raw_stg=sbin2dec_mat(raw_stg,[h,w]);
    raw=uint8(raw_stg);
end
```

分析：

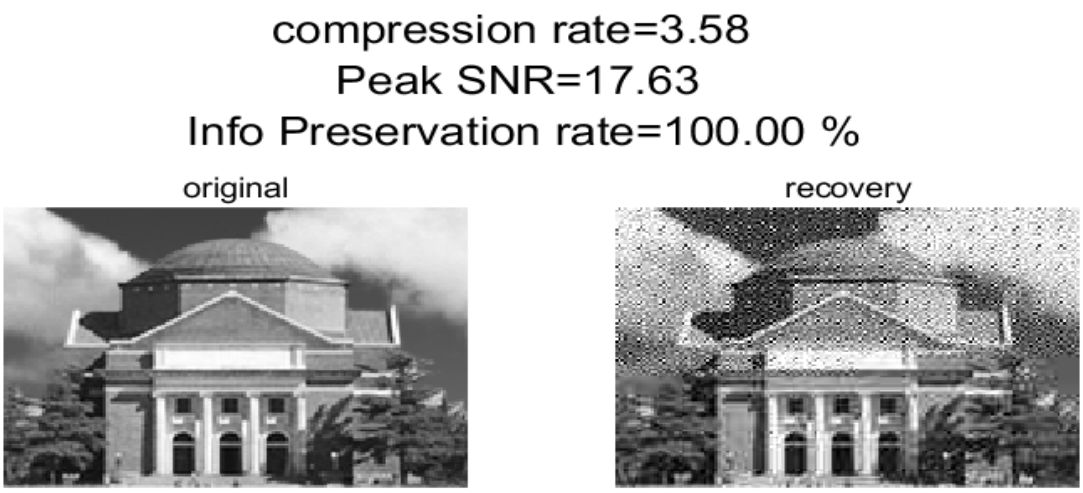
虽然空域方法能够保证很高的压缩率，而且能够将信息恢复，但是会导致图像恢复质量非常差，能被一眼发现出来。

附着信息容易被视为高频分量而舍弃。

2. 变换域方法

(a) 信息位逐一替换各个量化 DCT 系数最低位，再熵编码

效果:



关键设计细节:

关键技术问题在于:

1. 怎么把有符号十进制数矩阵表示为二进制矩阵

2. 怎么将二进制矩阵表示回有符号十进制矩阵

解决思路一：利用 MATLAB 的内置函数 bin2dec,dec2bin

这一思路的问题：将矩阵批量转换成二进制表示时，由于各个 entry 有正有负，且大小可以有很大差异，因此每个元素单独拿出来转换为二进制时的码长并不相同。matlab 为了将输出结果统一为 char 矩阵的形式，会自动为负数的补码表示向高位补'1'，正数的无符号表示向高位补'0'， 以将所有较小的二进制表示都延拓成等长的序列。

这里的"等长"，是与所有输入数据中的最长二进制码表示相同。对于输入图像而言，我们去直流之后各个元素分布在[-128,127]之间，因此码长至多为 8

因此使用 MATLAB 内置的 dec2bin 会将 signed int 图像矩阵转换为 H*W x 8 的用 char 表示的二进制矩阵。

修正思路一：干脆直接手写一个 sbin2dec： signed binary to decimal 函数，省的后面

手动转换

```
%convert char representation of a signed binary string
% to its decimal value.
function out = sbin2dec (x)
if  (x(1)=='1')
    out = bin2dec(x)-2 ^ (length (x));
else
    out = bin2dec(x);
end
```

遇到的新问题:

signed decimal →dec2bin→char arrays (each **row** correspond to a value)

sbin2dec: char array-> signed decimal

MATLAB 中将函数作为矩阵函数时，**惯例是按列**执行。但是我们解码时需要逐行对字符矩阵做解码操作。因此需要借助 arrayfun 手写一个**按行作用的矩阵算子**

```
d_flow=arrayfun(@(i) sbin2dec(b_flow(i,:)), 1:size(b_flow,1));
```

再做转换。

最后方案:

将上述两个操作封装起来定义一组 **sbin2dec_mat**

```
function DecM=sbin2dec_mat(CharM,sz)
d_flow=arrayfun(@(i) sbin2dec(CharM(i,:)), 1:size(CharM,1));
DecM=reshape(d_flow,sz);
end
```

并与 dec2bin 共同构成矩阵的 signed value/bin value 变换对:

signed dec matrix →**dec2bin** →char matrix → **sbin2dec_mat** →signed dec matrix

方法正确性测试：

```
>> M=randi([-128,127],[100,120]);
N=sbin2dec_mat(dec2bin(M),size(M));
error=norm(M-N);
>> disp(error)
     0
```

(b) 信息位逐一替换若干量化 DCT 系数最低位，再熵编码

效果：

---

compression rate=3.50
Peak SNR=18.26
Info Preservation rate=100.00 %

设计细节：

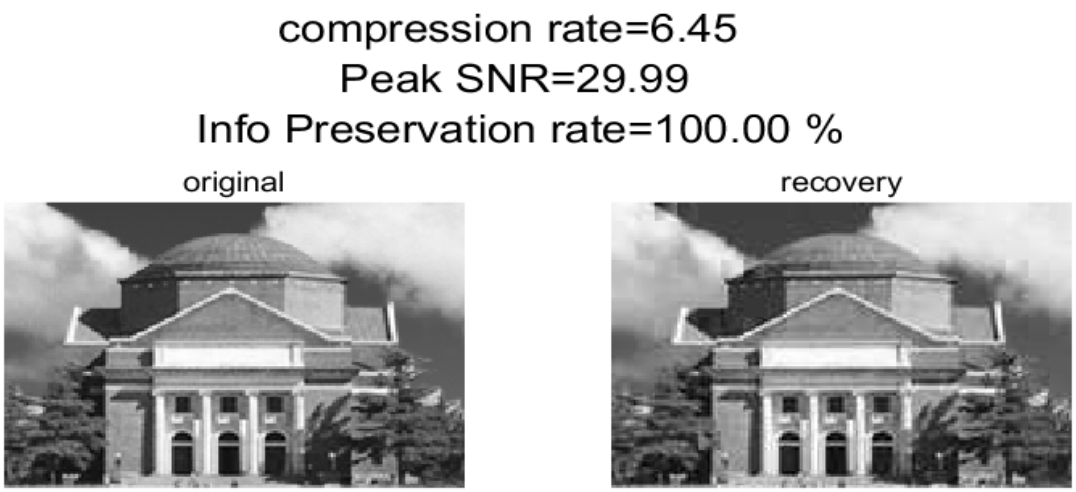
此时 information 不必与像素总数相当，可以少一些。将每个块中的部分像素的 LSB 替换成信息。**注意此方法与 a) 较为相似，实际上可以通过调节 b) 中信息的长度来还原为 a)**

```
elseif(tp==2)
    %C_q: 63 x ..
    B=cq_sz(2);
    %C_q_b: .. x 8 (binary char matrix.)
    C_q_b=dec2bin(C_q);
    %block-wise (64/) steganography.
    % number of blocks==B; number of elements hidden in a block=N.
    % length of info  ==L
    L=length(info);
    N=L/B;
    info_b=dec2bin(info);
    for blk=1:B
        i=(blk-1)*64;%start position of this block.
        j=(blk-1)*N ;%start position of the  region in info that will
be hidden in this block.
        C_q_b(i+1:i+N,end)=info_b(j+1:j+N); %remember we ONLY CHANGE
THE LSB!!
    end
    % get back to signed int matrix.
    C_q=sbin2dec_mat(C_q_b,cq_sz);
```

(C) 隐藏信息用 1,-1 的序列表示，再逐一追加在块 Zig-Zag 最后一个非零 DCT 系数之后;

如果原本该图像块的最后一个系数就不为零，那就用信息位替换该系数

效果：



评价：图像的复原效果好多了，说明这种方法更能保持高频分量的信息！

设计细节；

1. **信息**

```
%we require the length of info be as same as number of blocks.
    %note that info takes value in {-1,1}.
    %we first create {-1,0}-->y= -2x-1 -->{-1,1}
    info_pre=randi([-1,0],1,B);
    info_content=-2.*info_pre-1;
```

2. **隐写机制**

```
elseif(tp==3)
% good news: no binary conversion is included in the third method.
    %number of blks.
    B=h*w/64;
    % C_q: (64, B)
    %each columnn is a block(zigzagged). we write block-wise.
    for blk=1:B
        % find the non-zero elements in this column.
        col=C_q(blk);
        non_zero_ids=find(col);
        last_non_zero_id=max(non_zero_ids);
        % if last elem is not zero,
        %then last_non_zero_id===64, we hide in
        % the 64-th elem. otherwise, hide right after the last non-zero
        % elem. so...
        place_to_hide=min(last_non_zero_id+1,64);
        C_q(place_to_hide,blk)=info(blk);
    end
end
```

四：基于信号处理的人脸识别

1. 训练标准人脸特征向量

(a) 是否需要将训练样本图像大小归一化

不用。区域的特征向量定义中已经将区域的大小归一化掉了，只表示了区域内颜色出现频率的相对强弱。

The frequency that color n appears in region $\mathbf{R}$

$$f_n(\mathbf{R}) := \frac{1}{\#\,\mathbf{R}} \sum_{(x,y)\in\mathbf{R}} \mathbb{1}\{r_{x,y} << 2L + g_{x,y} << L + b_{x,y} = n\} \in [0,1]^{3\times L}$$

(b) 不同颜色集大小 L=3,4,5 之下，标准特征之间的关系

定性分析：

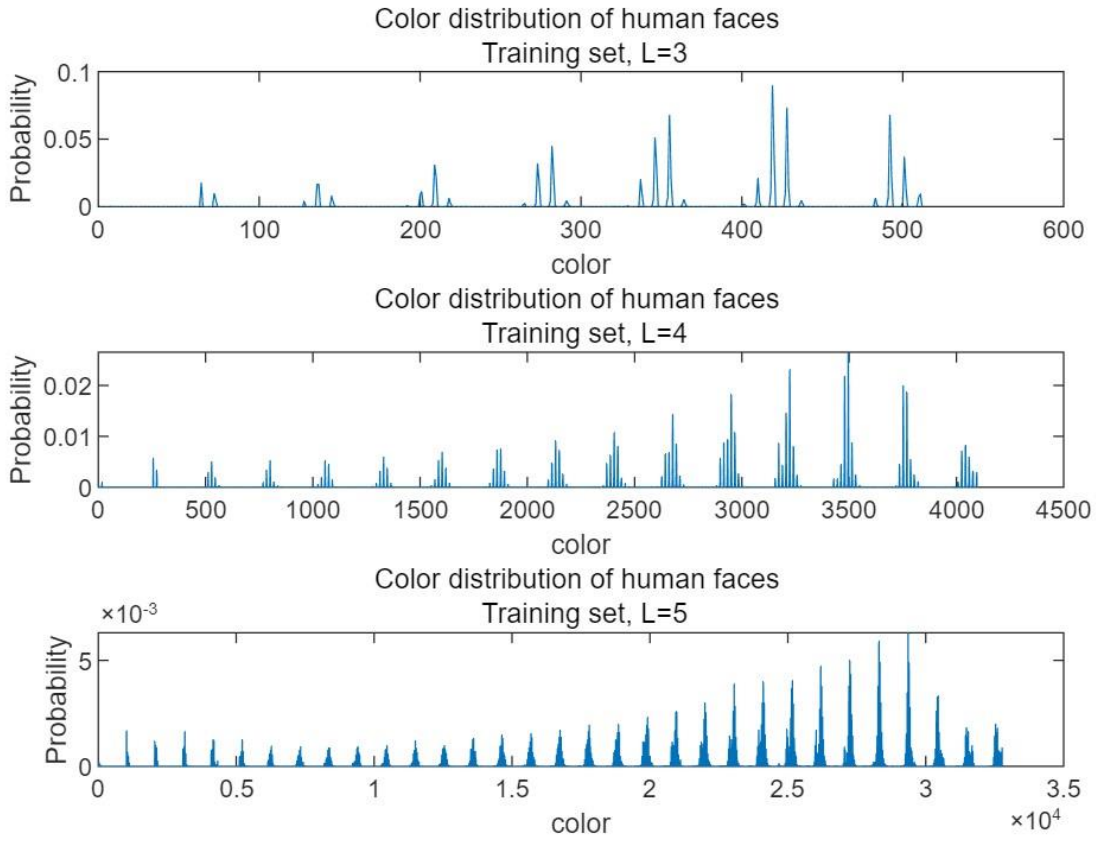
颜色集大小为$2^{3\times L}$ 因此 L 表征了颜色集的大小，也体现了当前颜色分类系统的分辨率。

L 越大，分辨率越高，对颜色的区分越细腻，类别更多，对噪声也越敏感，分类更加精准。

L 越小，分辨率越低，对颜色的区分越粗糙，类别更少，对噪声更迟钝，但是分类更加模糊

更高的 L 的分布谱，其峰峰值会合并成更低的 L 的分布谱的峰。

实验证据：



核心设计细节 1 ： 计算$\vec{u}(R)$

scale image representation with the representation of the color set together.

```
%scaling:
scale_ratio=2^(L-8);
%About scaling...
% The color of the image was represented in L=8 format:
% we have 2^{3 x 8} colors, each color is encoded as uint8
% r,g,b \in {0,1,...255} -----bin vector of 8 bits
```

```
%
% Now we want to linearly scale the color set and expand it to 2^{3xL}
% we should re-interpret the image and the colorset together.
% to reinterpret the color set, note that now each color is encoded
% by a bin vector of length L.
%
% similarly, we should also translate the color representation in the
% image from 8bit-world to L-bit world.
% if L>8, then we are not losing information, directly view x in 8-bit
% image as x*2^L/2^8 in L-bit world. the relative relation between
% different colors are linearly scaled.  e.g. (255)_8 --(65280)_16
% if L<8, then we are losing information to compress the
representation.
% e.g. (255)_8 --- (7.96875)_3 ~~7_3
%extract three components from the graph
r=floor(double(reshape(im(:,:,1),sz)).*scale_ratio);
g=floor(double(reshape(im(:,:,2),sz)).*scale_ratio);
b=floor(double(reshape(im(:,:,3),sz)).*scale_ratio);
```

核心设计细节 2：迭代方法计算$\vec{v}$

可以使用迭代方法计算均值。

数学推导详见附录 A2

```
v2=zeros(1,N);
for i=1:33
    dir="./Faces/"+string(i)+".bmp";
    im=imread(dir);
    %iterate the means
    v2=((i-1).*v2+u_R(im,L))./i;
end
```

这一方法等价于

```
for i=1:33
    dir="./Faces/"+string(i)+".bmp";
    im=imread(dir);
    %iterate the means
    v=v+u_R(im,L);
end
v=v./I;
subplot(1,2,1);
plot(v);
```

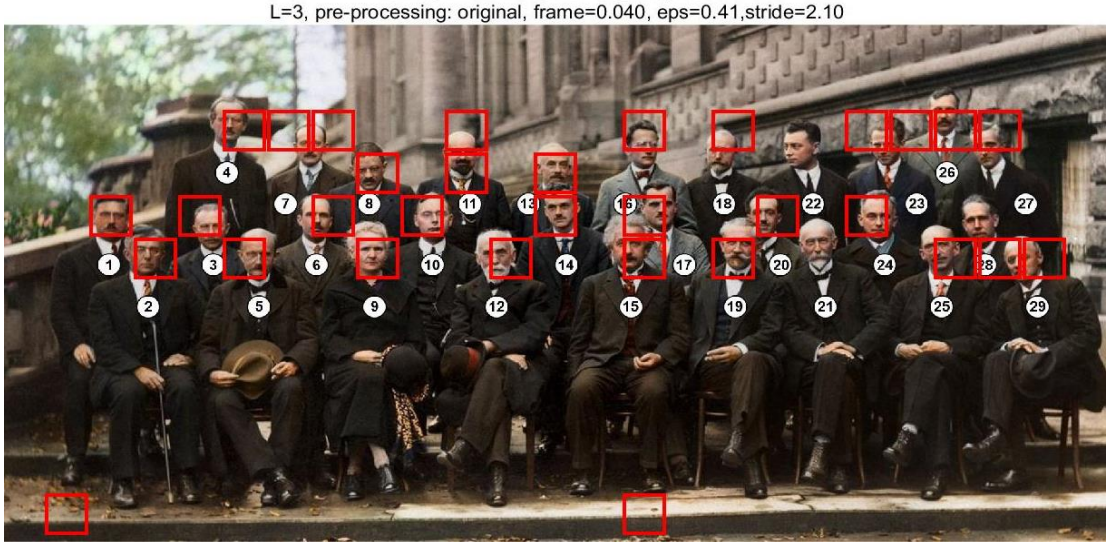
迭代方法是在 v 的各个 entry 可能非常大的情况下避免累加导致上溢出之下的优化算法，考虑到实际上 v 的各个 entry 不会高于 1，因此当训练集较小时后一种方法更加易读。

但是如果训练集很大则建议使用迭代而非累加。

2. 任意多人脸检测

结果：

Solvay Conference Facial Recognition
L=3, pre-processing: original, frame=0.040, eps=0.41,stride=2.10



Solvay Conference Facial Recognition
L=4, pre-processing: original, frame=0.040, eps=0.41,stride=2.10

Solvay Conference Facial Recognition
L=5, pre-processing: original, frame=0.040, eps=0.41,stride=2.10

评测效果对比分析：

其它超参数都不变，只改变 L=3,4,5，图像处理过程有这几点区别

1. 处理时间：随着 L 的增大， 图像处理时间明显增加。在 L=5 时消耗的远远高于 L=3，因为计算复杂度应该对 L 呈指数增长。
2. 灵敏度：

   L 越小，灵敏度越低，模型更加自信，找到的人脸总数多，但是误判也多，而且人脸定位并不精准。

   L 增大，分辨更加精细，但是模型更加保守，识别的人脸也更少，但是识别的位置和判定结果都更加精准。

评测效果整体分析：基本上较为成功。

在 L 较低时的误差来自于：

我选择的图像的环境色调与黄种人肤色类似，导致测试时出现了将手或大地误认为人脸的误判现象。这是这种训练方法的本质局限性导致的。

核心设计细节：

以一定的步长扫描整张图像的矩形区域，

计算区域的颜色特征向量，并度量其与标准人脸向量的相似度。

根据预先设定的敏感性阈值，判定此区域中的图像是否为人脸。

如果是人脸，则将人脸中心标记在此区域中

与此区域内其它可能的人脸中心相比较，取相似度最高的那个点，定义为此区域的中心

并消除此区域内的其它可能的人脸中心（局部聚类以减少重复）

如果不是人脸，则放弃此区域，再检查下一区域

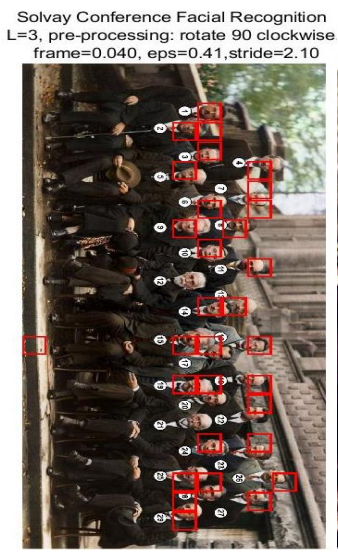
```
for x=H_f+1:H_s:H
    for y=W_f+1:W_s:W
        %the region that is currently being tested--"R"
        %its feature--"u", and difilarity with stdfeature v--"dif"
        R=im(max(x-H_f,1):min(x+H_f,H), max(y-W_f,1):min(y+W_f,W),:);
        u=u_R(R,L);
        dif=dis(u,v);
        %if detected.
        if (dif<eps)
            %select the possible region to draw a frame"frm"
            frm=frms(max(x-H_f,1):min(x+H_f,H), max(y-W_f,1):min(y+W_f,W));
            %leave a point in the center of the white canvas if no other
            %points are found in the same region.
            n_other=length(find(frm));
            if(n_other==0)
                frms(x,y)=dif;
            %if more than one point occurs in the same frame
            %pick the point with the smallest difference and select it as
            %the only centre of the frame and wipe other points out.
            else
                %xs, ys are arrays of the coordinates of the points in the frame.
```

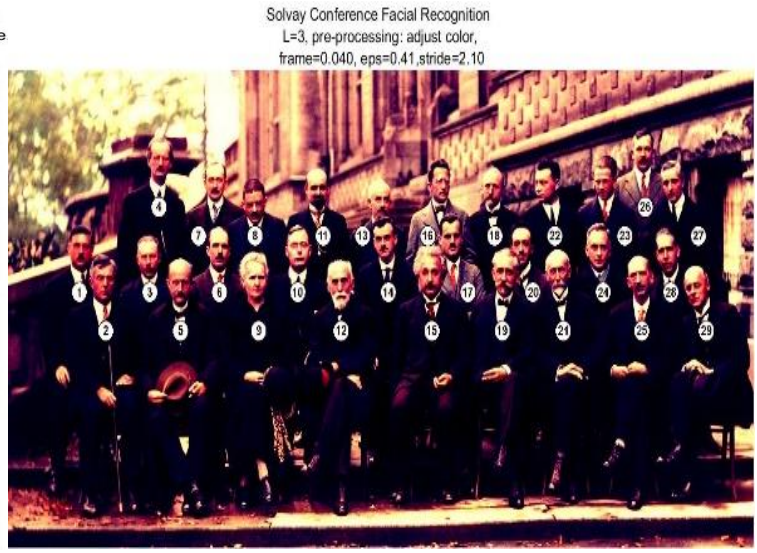
```
                [xs, ys, difs]=find(frm);
                %ctr_dif is the smallest difference of all points, while args are its linear ids.
                [ctr_dif,args]=min(difs);
                ctr_id=args(1);%pick one if tied
                frms(xs(ctr_id),ys(ctr_id))=ctr_dif;
            end
        end
    end
end
```

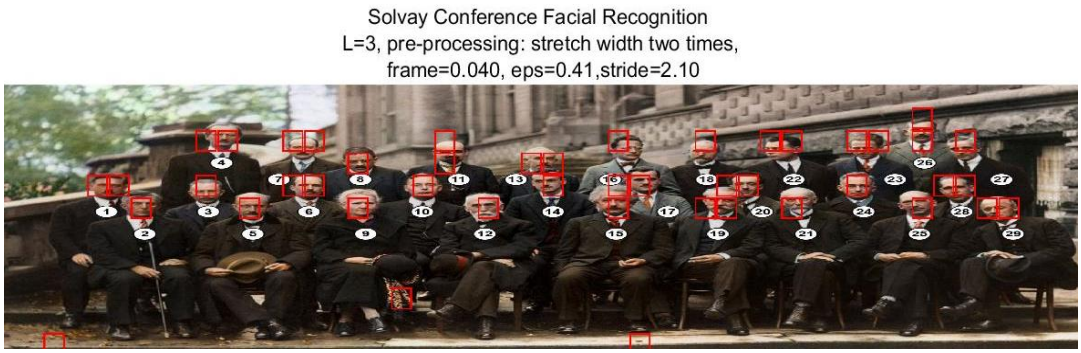
更多代码细节详见 test.mlx，此文件附有详细注释。

3. 图像预处理后的识别

效果：







结果分析：

1. **右旋 90°和横向拉长 2 倍**并未明显损害系统共工作性能，因为训练集中存在一些不同角度的人脸，而且特征向量抽取时进行过对区域大小的归一化。
2. **改变输入图像色调会严重降低系统工作性能**，因为会出现非常大的 distribution shift---训练集上的数据与测试集的数据的色彩分布有非常大的差异，而本算法的关键设计严重

依赖于（实际上是决定于）图像的色彩分布。因此，较为明显地改变图像色彩之后，本套系统设计会出现彻底失灵的现象，这是由系统算法设计的本质弱点导致的。

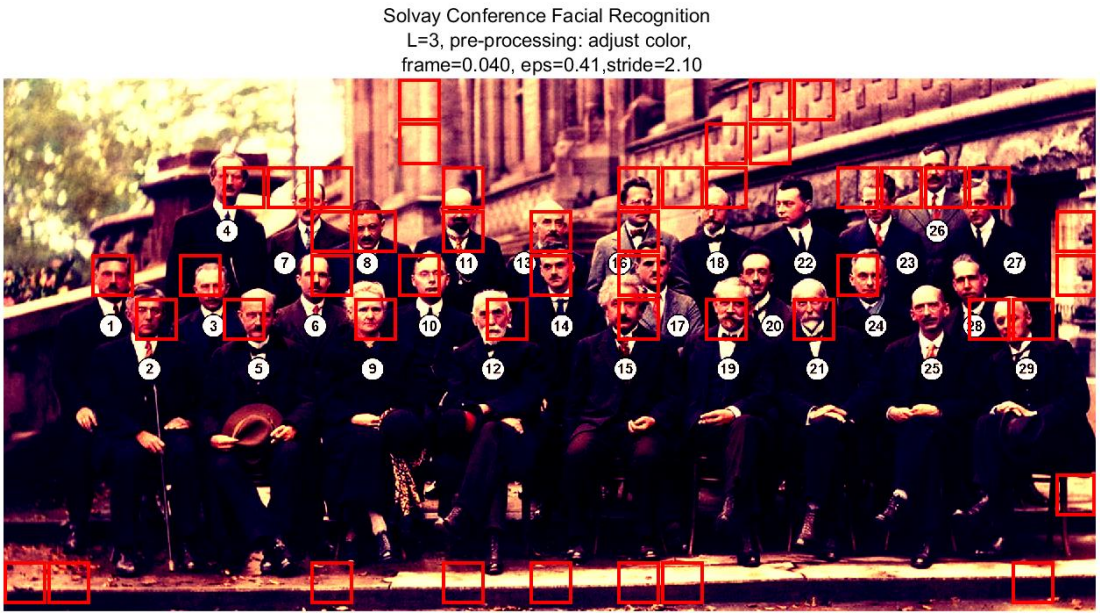
改进方法：

如果能增加训练数据集的多样性，则可以消除这种问题，比如训练时也过相同的滤镜：

```
%train
if synchronize_train_test==1
    v=train(L,"./Faces/",process_type);
else
    v=train(L,"./Faces/",0);
end
```

对训练集进行颜色修正后的某样本如右图

修正训练集后再进行识别，检测结果有了明显改进



此时的误差主要由于图像被漂染之后，背景色与人脸色更加接近，因此模型出现了更多的环境背景的误判。这是整体染色这种操作带来的后果。

设计细节

把图像过一个预处理模块再评测。另外也可以选择是否要对训练集进行相同处理后再测试。

```
% preprocessing
if (proc_tp==2)
    im=imrotate(im,-90);
elseif(proc_tp==3)
    sz=size(im);
    im=imresize(im,[sz(1),sz(2)*2]);
elseif(proc_tp==4)
    im=imadjust(im,[.2 .3 0; .6 .7 1],[]);
```

```
end
```

更多细节请见 prep.mlx

4. 如何选择人脸训练样本

1. 对样本进行分类，增大**类别数**和**每一类的样本数**
2. 增加人脸肤色的多样性，改变角度，光线，脸的朝向等，**特别是要增加人脸颜色与环境色相似的那些数据。**
3. 增加半身照片，或**人脸与脖子、肩膀、手臂等其它身体器官**共同出现的照片，让机器能学会分辨这些相似的色块

附录

A 数学推导

A1 直流组分的频域表示

Now we try to give a closed form of the frequency shift amount

$(D\mathbb{1}D^T)_{i,j} = \sum_{k\in[N]}(D\mathbb{1})_{i,.k}\cdot D^T_{k,j} = \sum_{k\in[N]}\sum_{l\in[N]} D_{i,l}\mathbb{1}_{l,k}D_{j,k} = \sum_{k,l\in[N]^2} D_{i,l}\cdot D_{j,k} = \sum_{l\in[N]} D_{i,l}\sum_{k\in[N]} D_{j,k}$

Before that, we need the following lemma:

Lemma :

$\forall w\neq 0, \forall \phi_0\in\mathbb{R} : \sum_{k=1}^{N} cos(kw+\phi_0) = \frac{sin(Nw/2)}{sin(w/2)}cos(\frac{N+1}{2}w+\phi_0)$

Proof :

With the help of following identities,

$$sinAcosB = (sin(B+A)-sin(B-A))/2$$

$$\begin{aligned} sinA-sinB &= sin(\frac{A+B}{2}+\frac{A-B}{2}) - sin(\frac{A+B}{2}-\frac{A-B}{2}) \\ &= 2\cdot sin(\frac{A-B}{2})cos(\frac{A+B}{2}) \end{aligned}$$

We conclude that

$\forall w\neq 0, \forall \phi_0\in\mathbb{R}$

$$\begin{aligned} \sum_{k=1}^{N} cos(kw+\phi_0) &= \frac{1}{sin(w/2)}\sum_{k=1}^{N} sin(w/2)cos(kw+\phi_0) \\ &= \frac{1}{sin(w/2)}\sum_{k=1}^{N}\frac{sin[(k+1/2)w+\phi_0]-sin[(k-1/2)w+\phi_0]}{2} \\ &= \frac{1}{2sin(w/2)}[sin((N+1/2)w+\phi_0)-sin((1/2)w+\phi_0)] \\ &= \frac{1}{2sin(w/2)}2sin(Nw/2)cos(\frac{N+1}{2}w+\phi_0) \\ &= \frac{sin(Nw/2)}{sin(w/2)}cos(\frac{N+1}{2}w+\phi_0) \end{aligned}$$

□

Main result

By the definition of DCT operator,

$$\mathbf{D} = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \cdots & \sqrt{\frac{1}{2}} \\ \cos\frac{\pi}{2N} & \cos\frac{3\pi}{2N} & \cdots & \cos\frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos\frac{(N-1)\pi}{2N} & \cos\frac{(N-1)3\pi}{2N} & \cdots & \cos\frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix} \quad D^{-1} = D^T$$

$$D_{i,j} := \sqrt{\frac{2}{N}} \begin{cases} \sqrt{\frac{1}{2}} & ,i = 1 \\ cos(\frac{(i-1)\pi}{2N} \cdot (2j-1)) & ,2 \le i \le N \end{cases}$$

$$= \begin{cases} \sqrt{\frac{1}{N}} & ,i = 1 \\ \sqrt{\frac{2}{N}} cos(\frac{(i-1)\pi}{N} j - \frac{i-1}{2N}\pi) & ,2 \le i \le N \end{cases}$$

with the help of the previous lemma, it is not hard to see that

$$\sum_{j=1}^{N} D_{i,j} = \sqrt{\frac{2}{N}} \cdot \begin{cases} \sqrt{\frac{N^2}{2}} & ,i = 1 \\ \frac{sin(N/2 \cdot \frac{i-1}{N}\pi)}{sin(\frac{i-1}{N}\pi/2)} cos(\frac{N+1}{2}\frac{i-1}{N}\pi - \frac{i-1}{2N}\pi) & ,2 \le i \le N \end{cases}$$

$$= \begin{cases} \sqrt{N} & ,i = 1 \\ \sqrt{\frac{2}{N}} \frac{sin(\frac{i-1}{2}\pi)}{sin(\frac{i-1}{2N}\pi)} cos(\frac{i-1}{2}\pi) & ,2 \le i \le N \end{cases}$$

In the sense of continuous extension$(\sum_{j=1}^{N} D_{0,j} := \lim_{i \to 0} \sum_{j=1}^{N} D_{i,j})$

we can make our expression more compact:

$$\sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{D_{i,j}} = \sqrt{\frac{\mathbf{2}}{\mathbf{N}}} \frac{\mathbf{sin}(\frac{\mathbf{i-1}}{\mathbf{2}}\pi)}{\mathbf{sin}(\frac{\mathbf{i-1}}{\mathbf{2N}}\pi)} \mathbf{cos}(\frac{\mathbf{i-1}}{\mathbf{2}}\pi), \forall \mathbf{i} \in [\mathbf{N}]$$

So the shift amount in frequency domain that corresponds to space-domain -128 operation can be expressed as

$$(D\mathbb{1}D^T)_{i,j} = \begin{cases} N & ,i = 1, j = 1 \\ \frac{2}{N} \frac{sin(\frac{i-1}{2}\pi)}{sin(\frac{i-1}{2N}\pi)} cos(\frac{i-1}{2}\pi) \cdot \frac{sin(\frac{j-1}{2}\pi)}{sin(\frac{j-1}{2N}\pi)} cos(\frac{j-1}{2}\pi) & ,i \ne 1, j \ne 1 \\ 2\frac{sin(\frac{j-1}{2}\pi)}{sin(\frac{j-1}{2N}\pi)} cos(\frac{j-1}{2}\pi) & ,i = 1, j \ne 1 \\ 2\frac{sin(\frac{i-1}{2}\pi)}{sin(\frac{i-1}{2N}\pi)} cos(\frac{i-1}{2}\pi) & ,i \ne 1, j = 1 \end{cases}$$

And the frequency shift is $C \to \tilde{C} := C - 128 D\mathbb{1}D^T$

A2 迭代方法计算均值

iteration

$\frac{S_n}{n} \to \frac{S_{n+1}}{n+1}$

$\delta_n = \frac{S_{n+1}}{n+1} - \frac{S_n}{n} = \frac{nS_n + na_{n+1} - nS_n - S_n}{n(n+1)} = \frac{n \cdot a_{n+1} - S_n}{n(n+1)} = \frac{a_{n+1}}{n+1} - \frac{S_n}{n(n+1)}$

$M_1 := a_1$

$M_{n+1} = M_n + \delta_{n+1} = (1 - \frac{1}{n+1})M_n + \frac{1}{n+1}a_{n+1} = \frac{n}{n+1}M_n + \frac{1}{n+1}a_{n+1} = \frac{1}{n+1}(nM_n + a_{n+1})$

Iterative method recap:

to calculate $M_I = \frac{\sum_{i=1}^{I} a_i}{I}$,we can execute the following algorithm

$M_0 := 0$

$M_n <= \frac{(n-1)M_{n-1} + a_n}{n} \quad \forall n \in [I]$

B 实验所用的源码文件夹之外的代码

B-1 对 JPEG 恢复效果的主观和客观评价

```
psnr=psnr(image,hall_gray);
subplot(1,2,1);
imshow(hall_gray);
title("original");
subplot(1,2,2);
imshow(image);
title("recovered");
```