**Matlab综合实验三 图像处理实验报告**

学号：2017011090

班级：无78

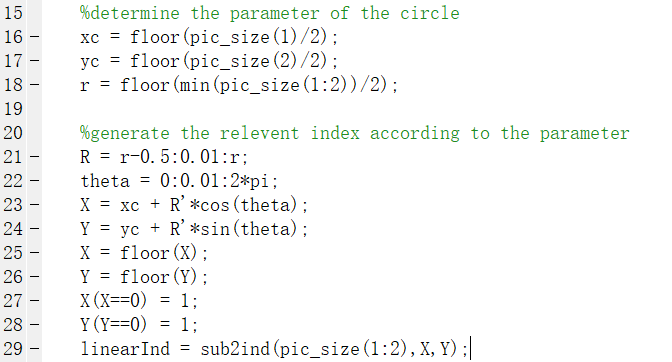
姓名：游子權

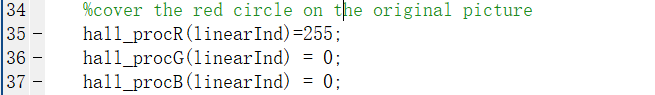
说明：在本作业第一章、第二章中，我皆以「用最少量的循环语句完成任务」为核心目标来设计程序。

1. **基础知识**

**(a)关键代码及方法如下：**

* q1\_2a.m







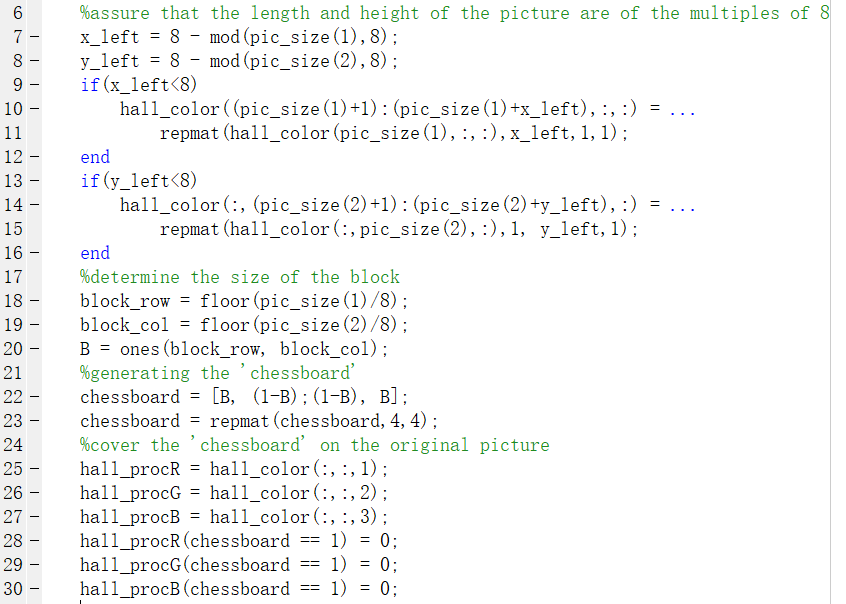
* 第16~18行：根据图片大小确定圆心位置与圆的大小。
* 第21~29行：利用圆的参数式，产生要染成红色的像素于图片矩阵中的行、列座标。并用sub2ind函式转成线性索引值（index）。
* 第35~37行：把上述的位置「染成红色」。



* 第44行：利用Matlab的Image I/O函数imwrite进行写入。
* **结果（hall\_withCircle.jpg）：成功，图片上出现红圈圈，惟其边缘锯齿状的现象较严重，是因为该图片本身就低清，离散化程度过高所致**。

**(b)关键代码及方法如下：**

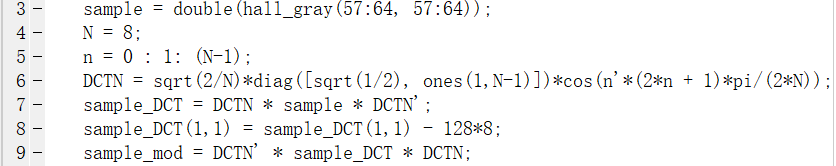
* q1\_2b.m



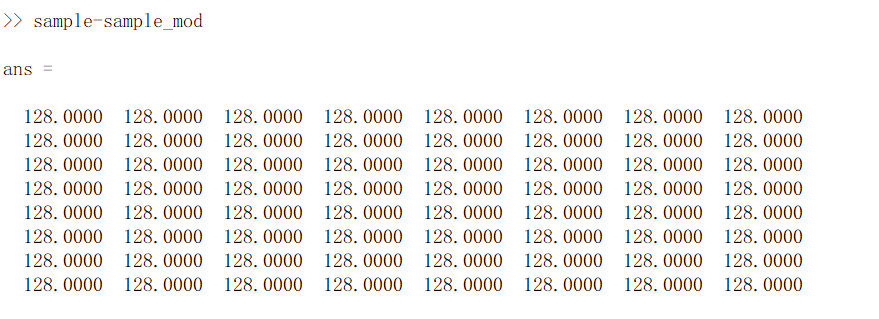
* 第7~16行：因为西洋棋棋盘大小为8×8，若能保证图片的长宽皆为8的倍数可方便划分「格子」。本人的做法是当图片不为8的倍数时，以其最后一行、最后一列的值填满至最接近的8的倍数的行、列大小。**本程序在之后第二章图像压缩编码中仍可沿用。因为其要求将图片分成若干个8×8的小块，故亦需保证长、宽为8的倍数。**
* 第18~20行：确定图像上每一「块」棋盘方格对应几行、几列。
* 第22~23行：重覆棋盘的最小单位以制作棋盘。
* 第25~30行：把做好的棋盘覆盖在原图上
* **结果（hall\_withChessboard.jpg）：成功**。

1. **图像压缩编码**
   1. 可以。二维离散余弦变换后的(1,1)元代表的是整张图片的直流分量。若我们是对N×N的图片块做二维DCT，且欲减去直流分量M，只需在其变换后系数矩阵的(1,1)元减去N×M即可。原因如下：

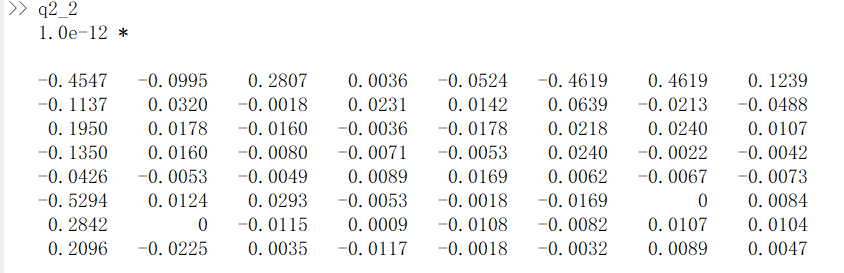
* **关键代码：q2\_1.m**



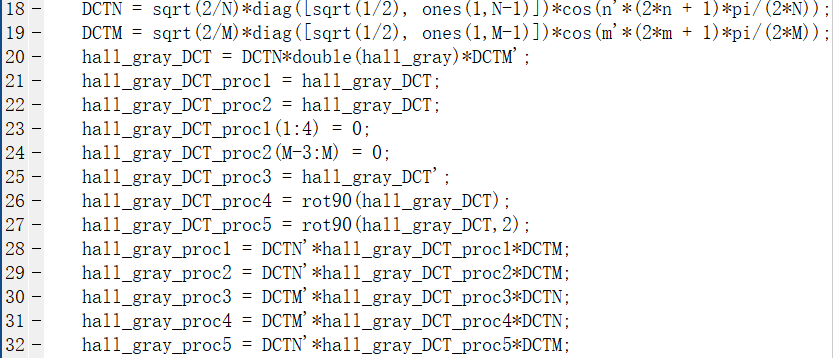
* 第6行：按照DCT的表达式生成对N维座标作DCT变换的矩阵。
* 第8行：如题要求所述，对8\*8的图片块，每个像素灰度值减去128，即相当于二维DCT系数的直流系数（第1行第1列）减去128×8。
* 第9行：对处理过的DCT系数做IDCT。以便比较逆变换后每个像素的灰度值是否皆比原本少128。
* **结果：成功。确实各像素灰度值皆比原本少128。**



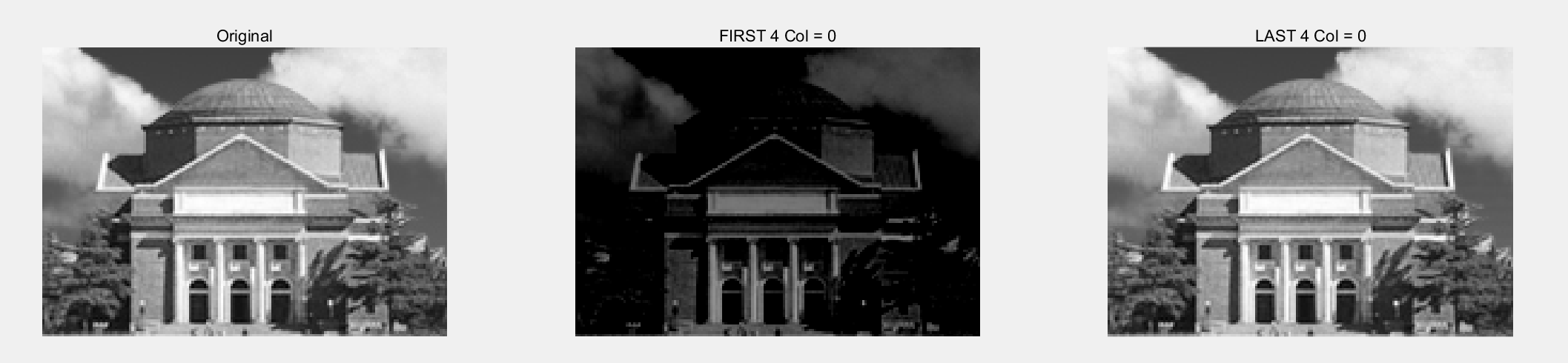
* 1. 本题的目标于上一题已经实现，本处仅比较与系统自带函数运算结果是否相同。
* **关键代码：q2\_2.m（代码附于附件中，此处从略）**
* 第7~9行：分别按自己矩阵计算，再按matlab的库函数计算，再将两者计算之差输出至命令行中。
* **结果：两者之差小于，可视为数值计算产生的舍入等误差。**



* 1. DCT系数(1)偏左上方为直流分量、(2)偏左下方为竖直方向快速振荡的分量、(3)偏右上方为水平方向快速振荡的分量、(4)偏右下方为水平、竖直方向皆快速振荡的分量。
* 将DCT后的系数前四列置0，影响的主要为如上所述的(1)(2)部分，直流分量变为0，使得处理后的图像黯淡许多；水平方向的快速振荡分量被置为0，但人眼的鉴别力有限，难以鉴别此所造成的差异。
* 将DCT后的系数的后四列系置0，影响的主要为如上所述的(3)(4)部分，但人眼的鉴别力有限，难以鉴别高频分量存在与否造成的差异。
* **本题关键代码与结果并于第4.题中说明。**
  1. 兹分述如下：
* 将DCT系数转置即相当于对原图像的灰度值矩阵转置。原因如下：
* DCT系数原本就是左上角密集（非0数值较多）、右下角稀疏，将DCT系数逆时针旋转90度，变成以左下角最为密集，而承3.(2)所述，偏左下方为竖直方向的高频分量，故经此一旋转后，可在图上观察到明显横条纹。有一种老电影画面的感觉。
* 同上所述，转180度后系数以右下角最为密集，而承3.(4)所述，偏右下方为水平、竖直两方向的高频分量，故经此一旋转后，可在图上观察到明显的网状条纹。有另一种老电影画面的感觉。
* **关键代码：q2\_3\_4.m**

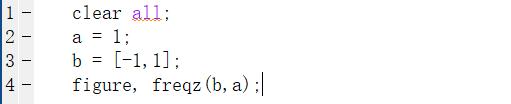


* 第18~19行：由于矩阵维度大小不一，需不同维度的DCT变换阵。
* 第21~27行：对于DCT系数阵做不同的处理。
* 第28~32行：将各个处理后的DCT系数逆变换。
* 第33~50行（从略）：将原图及处理后的各图绘在同一图窗中。
* **结果：如下图，与预期相符。**

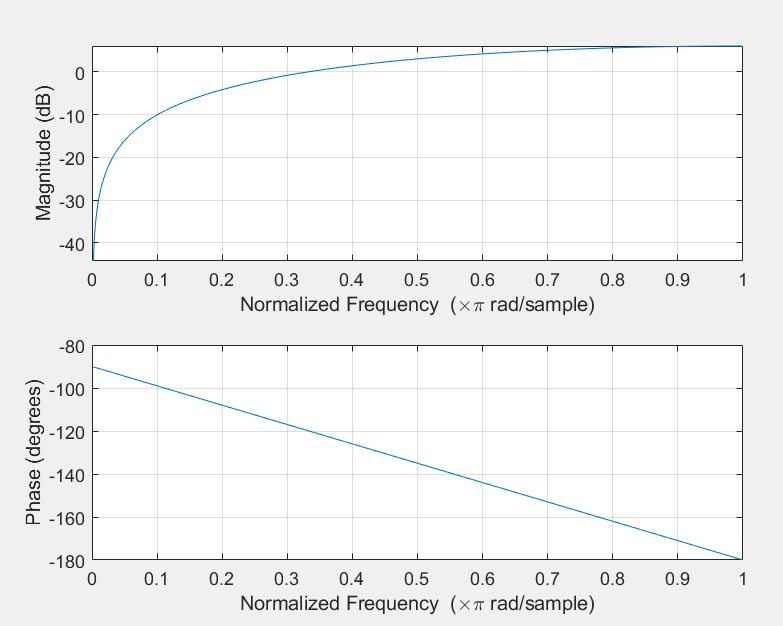




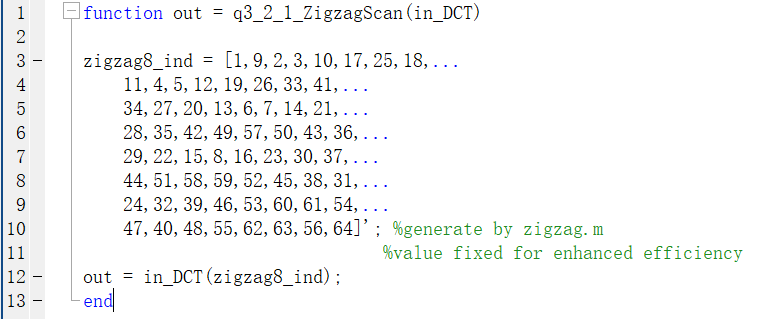
* 1. 如果认为差分编码是一个系统，则其输出与输入满足差分方程：，且。则可用Matlab中自带的函数分析其特性。
* **关键代码：q2\_5.m**



* 第4行：使用freqz函数求其频率响应。
* **结果：**上图为幅频响应，下图为相频响应。可见其**为一高通滤波器**。说明**DC系数的低频分量应该更多**。



* 1. 显然，设输入的数为，则给出该数所在的Category为
* **关键代码：getDCCategory.m（此处从略）**
  1. **思路：**Matlab擅于处理矩阵运算，故若能将Zigzag扫描路径所对应的线性索引（index）编列出来存至一数组中（下称「**顺序数组**」），让Matlab根据顺序数组中的索引在矩阵内依序取值，即可实现Zigzag扫描。对于一个8×8的矩阵，我们当然可以硬是用肉眼观察遍历的顺序编列出顺序数组，如下所示：
* **关键代码：q3\_2\_1\_ZigzagScan.m（本函数复用于本作业第三章第2题的第1种方法）**

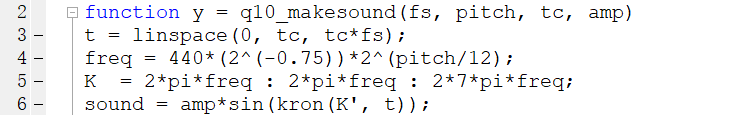


* 第3~10行：编列顺序数组。
* 第12行：根据顺序数组在矩阵中取值。

**但事实上我们并**

1. **基于Fourier级数合成音乐**
2. **关键代码与方法：**

* q10\_makesound.m



将第4问的q4\_makesound中，各次谐波的幅值不写死在函数中，改用一个1\*7的amp矩阵传入基波至第7谐波的幅值。

* q10\_song.m：将第8题计算出来的幅值引入，并调用q10\_makesound。

**结果**（q10\_eastRed.wav）：嗯……应该有像吧。

* 意犹未尽，我再次用同样的方法合成武陵高中和清华大学校歌。

**结果**（q10\_wuling.wav、q10\_tsinghua.wav）：果然，歌曲一长就听出端倪了，低音的时候确实蛮像的，但当到了高音，其更高频的谐波分量听起来显得很不自然，依照我们的生活经验与对乐器的主观想像，高音的时候不致再有很高频的谐波了。

1. **关键代码与方法：**

* q9/ makeWulingSong.m：输入由第9.题产生的FREQ\_INFO矩阵，对于每个音，会自动挑选FREQ\_INFO中最接近的频率的幅值矩阵，调用makesound\_harmonic.m合成乐音，输出《武陵高中校歌》。
* q9/makesound\_harmonic.m： 将第4问的q4\_makesound中，各次谐波的幅值不写死在函数中，改用可变大小的amp矩阵传入基波至第n谐波的幅值，并据以合成音乐。
* **结果（output\_wuling.wav）**：还行吧，我不希望它再更像了。

1. **关键代码与方法：**

* q12/ musicAnalysisGUI.mlapp：本GUI是用Matlab R2019a的App Designer所设计，与GUIDE有所不同。请以Matlab R2019a方可开启。
* **结果：如下页图所示。**



**▲图12-1：Matlab App Designer介面**



**▲图12-2：设计GUI可成功运行**