

信号与系统－ MATLAB 综合实验之图像处理¹

谷源涛

gyt@tsinghua.edu.cn

二〇一五年七月六日

¹本文为清华大学电子工程系三字班本科生课程“MATLAB 高级编程与工程应用”的教学课件。请勿转载和过度引用。

目录

第一章 基础知识	1
第一节 数字图像	1
第二节 彩色图像	2
第三节 练习题	3
第二章 图像压缩编码	4
第一节 引言	4
第二节 信源编码	4
第三节 静止图像编码标准—JPEG	5
2.3.1 概述	5
2.3.2 编码器	6
2.3.2.1 分块	6
2.3.2.2 离散余弦变换 (DCT)	6
2.3.2.3 量化	9
2.3.2.4 熵编码	10
2.3.3 解码器	13
2.3.3.1 熵解码	13
2.3.3.2 反量化	14
2.3.3.3 DCT 逆变换	14
2.3.3.4 拼接	14
2.3.4 码流组织	14
2.3.5 补充知识	14
第四节 练习题	15
第三章 信息隐藏	18
第一节 背景知识	18
第二节 空域信息隐藏技术	18
第三节 DCT 域信息隐藏技术	19
第四节 练习题	19

第四章 人脸检测	20
第一节 背景知识	20
第二节 基于彩色直方图的人脸检测方法	20
4.2.1 选取特征	20
4.2.2 训练标准特征	22
4.2.3 检测	22
第三节 练习题	22

第一章 基础知识

本章介绍计算机存储和处理图像的基础知识。

第一节 数字图像

图像是光怪陆离的花花世界在某平面上的投影信号。该信号一般在连续空间上定义，其幅度也连续变化，所以自然界的图像大多是模拟信号。

但也有空间上离散的图像，如图 1.1 所示：检测色盲用的图案由一块块不同颜色的、有间隔的斑点构成；“麦田怪圈”是在天空俯视麦田看到的奇怪符号，貌似连续的图形其实是由一个个离散的麦秆和麦穗构成的。第一个例子说明人眼有表现为低通滤波器的强烈意愿，我们很容易放弃高频分量；第二个例子则证明即使愿意人眼也看不到频率太高的分量，即眼睛的空间分辨力有限。



图 1.1：离散图像示例

公元二零一一年春天，一场盛大的晚会将万众瞩目的清华大学百年校庆推向高潮。晚会所用大屏幕（图 1.2）也是离散图像的典型代表：远距离看起来连续变化的图案其实是由间距很大的发光管组成的。

图像信号的幅度也可以离散。曾经流行过一个能够“评价显示器质量”的图片（图 1.3）。这说明纵然亮度是连续变化的，我们仍可以在不影响主观感受的条件下，对图像信号强度进行离散化处理。

现在我们即可解释数字图像的合理性。所谓数字图像，就是用空间离散（用二维矩阵表示）幅度也离散（一般取值 0 ~ 255）的二维数字信号表示的图像。如数字灰度图像（图 1.4）所示大礼堂房檐的一个 8×8



图 1.2: 清华大学百年校庆晚会大屏幕



图 1.3: 量化图像示意

块被放大后的样子，其对应存储在计算机中的数据是一个矩阵 \mathbf{P}

$$\mathbf{P} = \{P_{x,y}\} = \begin{bmatrix} 79 & 80 & 85 & 99 & 121 & 128 & 128 & 132 \\ 78 & 81 & 87 & 106 & 125 & 134 & 118 & 117 \\ 80 & 81 & 88 & 97 & 158 & 198 & 121 & 113 \\ 88 & 88 & 84 & 95 & 217 & 221 & 123 & 108 \\ 137 & 130 & 70 & 90 & 237 & 233 & 125 & 103 \\ 160 & 151 & 66 & 74 & 232 & 247 & 130 & 96 \\ 164 & 157 & 85 & 56 & 219 & 254 & 134 & 91 \\ 171 & 171 & 98 & 48 & 204 & 254 & 146 & 87 \end{bmatrix} \quad (1.1)$$

其中 0 表示黑色（最暗），255 表示白色（最亮）。

第二节 彩色图像

图像大多是彩色的。色彩有多种表示方式，如 RGB 表示，即分别用三个 $0 \sim 255$ 的数代表红绿蓝三基色的强度。还有 YCbCr 表示，Y 即亮度（Luminance），Cb 和 Cr 分别表示蓝色度和红色度。所谓色度（Chrominance），即任意一种颜色与亮度相同的指定的参考色之间的差异。两种表示方法的对应关系是

$$Y = 0.299R + 0.587G + 0.114B \quad (1.2)$$

$$Cb = -0.169R - 0.331G + 0.500B + 128 \quad (1.3)$$

$$Cr = 0.500R - 0.419G - 0.081B + 128 \quad (1.4)$$

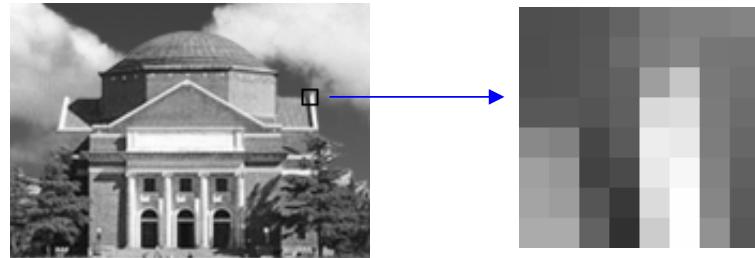


图 1.4: 数字灰度图像

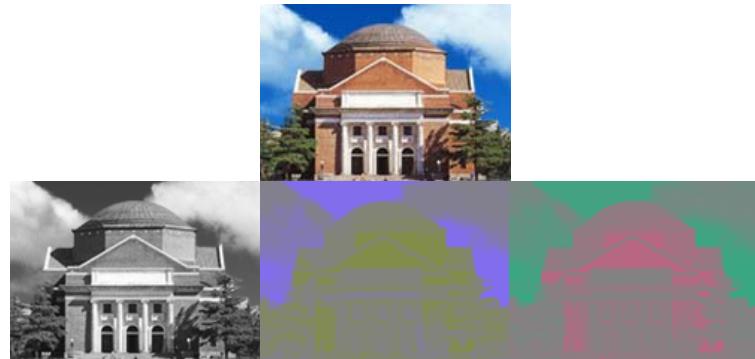


图 1.5: 数字彩色图像

所以彩数字图像需要用三个矩阵描述，分别代表三基色或者亮度和红蓝色度，如图 1.5 所示。换句话说，每个像素点对应一个三元组

$$P_{x,y} = [R_{x,y}, G_{x,y}, B_{x,y}] \quad (1.5)$$

第三节 练习题

在 MATLAB 中，像素值用 uint8 类型表示，参与浮点数运算前需要转成 double 型。本章练习题中“测试图像”指的是 hall.mat 中的彩色图像。

1. MATLAB 提供了图像处理工具箱，在命令窗口输入 help images 可查看该工具箱内的所有函数。请阅读并大致了解这些函数的基本功能。
2. 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。

第二章 图像压缩编码

本章以经典图像编码标准 JPEG 为对象，练习 MATLAB 编程中的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。通过本章学习，希望同学们掌握

- JPEG 标准的基本原理；
- 变换域编码和量化的基本思想；
- MATLAB 处理矩阵和图像的常用命令。

第一节 引言

首先我们计算一幅图像所需的存储空间。假设一幅彩色图像，高和宽的像素数分别是 H 和 W ，每个像素的色彩分别用 RGB 三个分量表示，每个分量 8 bits（取值范围 $0 \sim 255$ ），即存储每个像素需要 24 个 bits。这样的一幅图像总共需要 $24HW$ bits。具体的，对一个只有两百万像素的数码相机/手机来说，其摄像头分辨率是 $1600 \times 1200 \approx 1.92$ Mega-Pixel，为存储其拍摄的一张相片就需要

$$S = 24 \times 1600 \times 1200 = 46080000 \text{ bits} \approx 46 \text{ Megabits} \approx 5.8 \text{ Megabytes} \quad (2.1)$$

的存储空间。实在太大了，因而有必要对图像进行压缩编码以方便存储。

压缩编码后的图像不仅少占用空间，在同样信道条件下还可以减小传输时间。因而是通信学科的重要研究领域，和语音编码、视频编码一起，称为信源编码¹。

第二节 信源编码

信源编码就是把待传输的源信号（符号）映射成码字（符号），再送上信道传输（或者存储）的过程。比如著名的莫尔斯电码，就是将二十六个英文字母和十个数字及空格等符号映射成点和横两种符号的组合，如表 2.1 所示。其中用短符号代替出现频率高的字母如 e、T 等，用长符号代替出现频率低的字母，从而减小了编码后总的符号长度。

在用短码字代表出现频率高的编码方式中最有代表性的是 Huffman 编码。可证明当信源符号频率是 2 的负整数幂时，Huffman 编码最优（平均码长最短）。此外它还是前缀码，译码比较简单。本章 2.3.2.4 将讲到用这种方法进行 DCT 系数编码²。

¹本章出现的“编码”指“压缩编码”，实际上编码还有增加冗余（如信道编码）、加密等多种功能。

²数据结构等课上已经讲过，本章不再详细介绍。

表 2.1: 莫尔斯电码表

字符	码表	字符	码表	字符	码表
A	-	N	-·	1	-----
B	-··	O	---	2	-··-
C	-·-	P	-··	3	-····
D	-··	Q	-··-	4	-···-
E	.	R	-·	5	-····
F	-··-	S	··	6	-·····
G	-··-	T	-	7	-·····
H	···	U	-·-	8	-·····
I	..	V	··-	9	-····-
J	-··-	W	-··	0	-····-
K	-·-	X	-··-	.	-····
L	-··-	Y	-··-	?	-····
M	--	Z	-··-	/	-····-

如果先编码再解码后恢复的信号和原信号完全相同，即压缩过程没有信息损失，则称为熵编码³或无损编码。上面提到的莫尔斯电码和 Huffman 编码都是熵编码。常用的 Zip 和 WinRAR 等数据压缩软件也属于熵编码，其核心是 LZW 算法。

熵编码的缺点是一般压缩比不高，同学们可以找些 txt 文件，用 WinRAR 进行压缩，可发现压缩比一般是 $3 \sim 5$ ⁴。为了提高压缩比，对语音和图像等数据量更大且对失真要求不严格的信源一般采用有损编码。

有损编码和熵编码相反，其解码重建的结果和原信号不同。对数据来说，这个不同（失真）可能是灾难性的。但对语音和图像、视频等信源而言，只要不影响主观感受，用户并不太在意重建信号是否和原信号完全相同⁵。因而信源编码一般指有损压缩编码，即研究在一定的失真容限下大幅度提高压缩比的方法。

第三节 静止图像编码标准—JPEG

2.3.1 概述

上个世纪八十年代中期，国际电联（ITU）和国际标准化组织（ISO）联合成立专家组共同制定了第一个连续调（多灰阶）静止图像压缩标准⁶，并以联合图像专家组（Joint Photographic Experts Group, JPEG）进行命名。

1988年 JPEG 选定了基于 DCT 的变换域编码方法；到 1990 年完成了模拟、测试和文档化算法工作；1991 年公布了标准草案；1992 年发布了正式的国际标准，此后又对标准做过扩展，最新的 JPEG 2000 标准于 1997-2000 年间制订。

受目的和篇幅所限，本节介绍的内容是 JPEG baseline 的核心子集，即只包括静止灰度图像基于 DCT

³熵的概念将在微机原理、通信原理、随机过程、信息论等一系列课程中讲解。

⁴也存在压缩比小于 1 或者大于 100 的 txt 文件，你能构造出来吗？

⁵研究表明感觉加权等技术能使失真的语音听着更真实。

⁶在此之前的数据压缩标准只是针对黑白二值图像，面向传真等应用。

的顺序编码模式，而不涉及彩色图像、逐帧压缩图像（Motion JPEG）、无损编码（Lossless coding）、累进扫描（Progressive scan）、分层（hierarchical）等模式或选项。

2.3.2 编码器

编码器结构如图 2.1 所示。主要包括分块、离散余弦变换、量化和熵编码等几部分。流程如下：首先将图像分割成同样大小的小方块，然后逐一对每块做二维离散余弦变换，接下来用不同的步长量化变换系数，再对量化后的系数进行熵编码。下面逐一详细介绍。

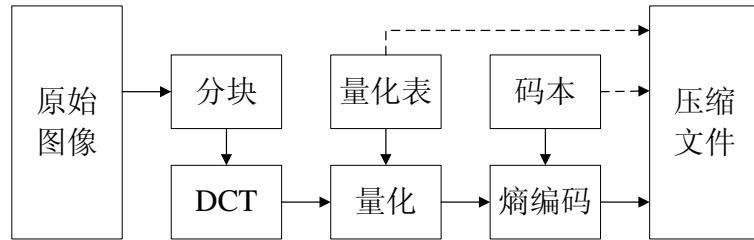


图 2.1: 编码器结构框图

2.3.2.1 分块

由于待处理的图像尺寸是任意的，为便于统一处理，首先进行分割操作。图像被分成 8×8 的若干个不重叠的小图像块，后续操作均针对小图像块进行。处理顺序是从左上角开始，先行再列依次编码各个块，如图 2.2 所示。⁷

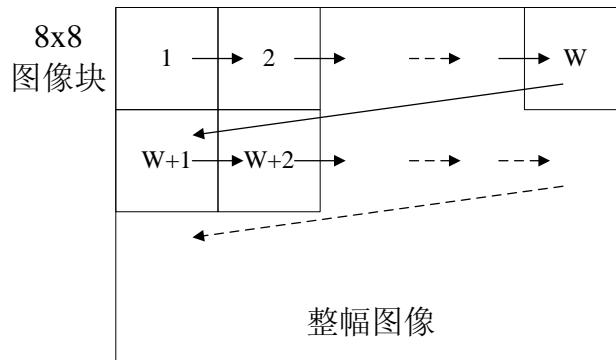


图 2.2: 分块及扫描顺序

进行后续操作前还要对图像进行预处理，即将每个像素的灰度（亮度）值减去 128⁸。

2.3.2.2 离散余弦变换 (DCT)

DCT 和傅里叶变换很像。我们已经学过

- 傅里叶级数 (FS) 是连续时间周期信号到无限长离散 (谐波幅度) 序列的一一映射；

⁷如果图像尺寸不是 8 的整倍数，则将最右侧的列/最下方的行内的元素向右/下延伸图像，直至达到 8 的整倍数为止。你知道这样处理的原因吗？

⁸本来 8 bits 的表示区间是 0 ~ 255，减去 128 后变成了 -128 ~ 127，想想为什么？

- 傅里叶变换 (FT) 是连续时间信号到到连续频率谱的一一映射；
- 离散时间傅里叶变换 (DTFT) 是离散时间序列到连续频率周期谱的一一映射；

同理，我们可以推测存在一种

- 离散傅里叶变换 (DFT) 是离散时间周期序列到离散频率周期谱的一一映射；

和连续时间傅里叶变换类似，离散傅里叶变换用复指数序列对时域序列进行展开，其展开系数（即变换结果）也是复数。为了避免出现复数，我们丢掉复指数中的正弦分量，只用余弦做基底，这就是离散余弦变换⁹。

首先学习一维 DCT。长度为 N 的矢量 $\mathbf{p} = [p_0, p_1, \dots, p_{N-1}]^T$ 可以用 N 个余弦序列加权表示为

$$\mathbf{p} = \sum_{i=0}^{N-1} c_i \cos \left[\begin{array}{c} 1 \\ 3 \\ \vdots \\ 2N-1 \end{array} \right] \frac{i\pi}{2N} \quad (2.2)$$

其中 $\{c_i\}$ 是 \mathbf{p} 的 DCT 系数，

$$c_i = \alpha_i \sum_{x=0}^{N-1} p_x \cos \frac{i(2x+1)\pi}{2N}, \quad \forall i = 0, 1, \dots, N-1 \quad (2.3)$$

式 (2.3) 中常数

$$\alpha_i = \begin{cases} \sqrt{\frac{1}{N}} & i = 0 \\ \sqrt{\frac{2}{N}} & \text{elsewhere} \end{cases} \quad (2.4)$$

由（时域或空域）矢量 \mathbf{p} 到变换域矢量 $\mathbf{c} = [c_0, c_1, \dots, c_{N-1}]^T$ 的过程称为 N 点 DCT。作为线性变换，DCT 可以用矩阵算子形式描述为

$$\mathbf{c} = \mathbf{D}\mathbf{p} = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \cdots & \sqrt{\frac{1}{2}} \\ \cos \frac{\pi}{2N} & \cos \frac{3\pi}{2N} & \cdots & \cos \frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(N-1)\pi}{2N} & \cos \frac{(N-1)3\pi}{2N} & \cdots & \cos \frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix} \quad (2.5)$$

\mathbf{D} 表示 DCT 算子，是 $N \times N$ 矩阵。

DCT 是可逆的，即在 \mathbf{p} 所在的时域或空域和 \mathbf{c} 所在的 DCT 域存在一一映射。由 DCT 系数 \mathbf{c} 到 \mathbf{p} 的逆过程称为 IDCT。注意 \mathbf{D} 是酉矩阵，DCT 是正交变换，所以有

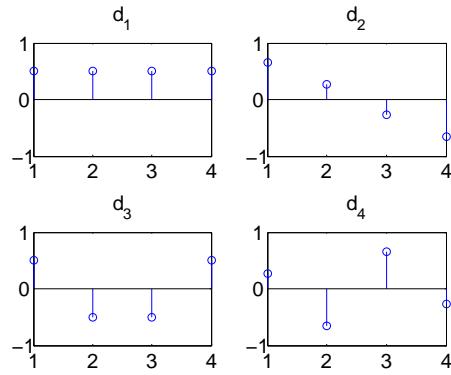
$$\mathbf{p} = \mathbf{D}^{-1}\mathbf{c} = \mathbf{D}^T\mathbf{c} \quad (2.6)$$

图 2.3 和 2.3 分别画出了 $N = 4$ 和 16 两种情况下一维 DCT 变换的基底。

如果待处理的是二维信号，比如待处理的图像

$$\mathbf{P} = \{P_{x,y}\} = [\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{N-1}] \quad (2.7)$$

⁹ N 点 DCT 系数可由 $2N$ 点 DFT 系数得到。郑版《信号与系统》下册 163 页有详细推导，数字信号处理课上将重点讲解 DFT 和 DCT。

图 2.3: $N=4$ 时一维 DCT 变换的基底

其中 $P_{x,y}$ 表示 (x, y) 位置像素点的亮度值， \mathbf{p}_k 表示第 k 列像素点的值。我们可以先对像素矩阵逐列做一维 DCT，再对变换后的系数矩阵逐行做一维 DCT，上述过程就是二维 DCT。用矩阵描述为

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

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

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

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

注意式 (2.8) 表示依次对各行做 DCT，并将变换结果组成新的矩阵

$$[\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{N-1}] \mathbf{D}^T = \left[\mathbf{D} [\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{N-1}]^T \right]^T = \left[\mathbf{D} \begin{bmatrix} \mathbf{c}_0^T \\ \mathbf{c}_1^T \\ \vdots \\ \mathbf{c}_{N-1}^T \end{bmatrix} \right]^T \quad (2.12)$$

所以二维 DCT 系数的表达式是

$$C_{i,j} = \alpha_i \alpha_j \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P_{x,y} \cos \frac{i(2x+1)\pi}{2N} \cos \frac{j(2y+1)\pi}{2N} \quad \forall i, j = 0, 1, \dots, N-1 \quad (2.13)$$

其意义是任意 $N \times N$ 矩阵 \mathbf{P} 都可由 $N \times N$ 个二维余弦序列（矩阵）加权表示，即

$$\mathbf{P} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_{i,j} \mathbf{D}^{(2)}(i, j) = \mathbf{D}^T \mathbf{C} \mathbf{D} \quad (2.14)$$

其中 $\mathbf{D}^{(2)}(i, j)$ 表示二维 DCT 的第 (i, j) 个基矩阵，其第 (x, y) 个分量为

$$\mathbf{D}_{x,y}^{(2)}(i, j) = \alpha_i \alpha_j \cos \frac{i(2x+1)\pi}{2N} \cos \frac{j(2y+1)\pi}{2N} \quad (2.15)$$

所以二维 DCT 和 IDCT 可分别表示为

$$\mathbf{C} = \mathbf{D}\mathbf{P}\mathbf{D}^T \quad \mathbf{P} = \mathbf{D}^T \mathbf{C} \mathbf{D} \quad (2.16)$$

也可表示为

$$\mathbf{C} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P_{x,y} \mathbf{D}^{(2)}(x, y) \quad \mathbf{P} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_{i,j} \mathbf{D}^{(2)}(i, j) \quad (2.17)$$

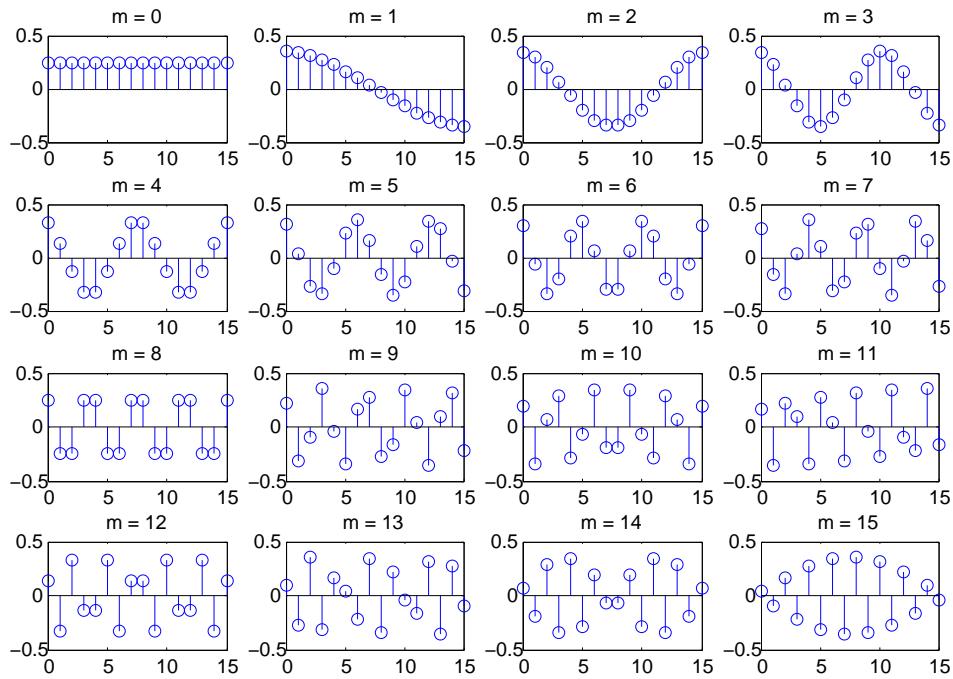
图 2.4: $N=16$ 时一维 DCT 变换的基底

图 2.5 画出了 $N = 4$ 时二维 DCT 变换的基底。

JPEG 对每个图像块应用 8×8 DCT 变换。不同位置的 DCT 系数的大小反映了图像亮度值的变化快慢。比如左上角的系数 $C_{0,0}$ 称为直流 (DC) 分量，表示图像块的整体亮度；其他系数称为交流 (AC) 分量，其中右上方的系数反映了图像块中横向变化的纹理的强度，同理，右下方的系数表征横竖两个方向都迅速变化的成分。由于常见的景观和相片等图像以缓慢变化为主，因而 DCT 系数矩阵的左上方取值较大，越向右下方取值越小。

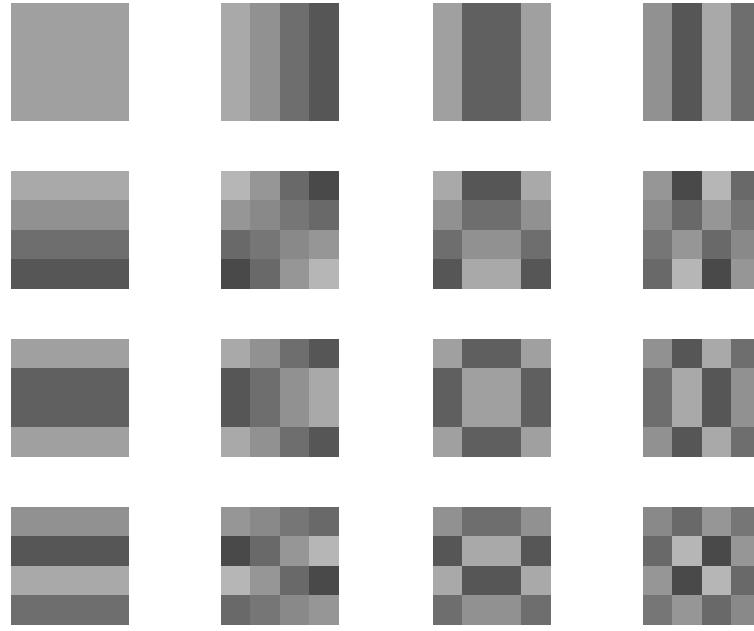
对每个图像块进行的 8×8 DCT 变换将 64 个像素灰度值变成了相同数量的 DCT 系数，似乎我们没捡到什么便宜，真是如此么？当然不是。像素值是一个也不能少，但 64 个 DCT 系数却不必享受平等待遇。为了压缩存储空间，我们必须抛弃掉一些分量，无疑右下角的“小”数是最先被丢弃的部分，这就是应用 DCT 进行压缩编码的原因——虽然它并没有实现压缩，但为减少数据量做足了准备。

2.3.2.3 量化

JPEG 编码能实现图像压缩的关键步骤是对 DCT 系数进行量化。因为人眼对图像中大块的连续亮度和色彩最为敏感，对快速变化的细节部分却不太关注¹⁰。所以允许以不同的失真处理 DCT 系数。

首先对每个 DCT 系数 $C_{i,j}$ 分别定义一个量化步长 $Q_{i,j}$ ，因而量化矩阵和系数矩阵大小相同，也是 8×8 。量化矩阵元素可以是 $1 \sim 255$ 的整数值。所有元素都为 1 的量化矩阵对应准无损压缩（失真仅来自

¹⁰当然不是一贯如此，比如海港上空航拍到的潜水艇和某影星脸上没被粉底盖住的雀斑，只是一个小点却可能颇受关注，所以 JPEG 有其应用的局限性。但在大多数情况下，高频失真是可以接受的。

图 2.5: $N=4$ 时二维DCT变换的基底

计算的舍入误差)。量化步长越大, 压缩比越高, 失真也越大。JPEG 中亮度信号 DCT 系数的量化系数是

$$\mathbf{Q} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (2.18)$$

例2.1. 已知某图像块 DCT 系数矩阵中直流分量 $C_{0,0} = 100$, 求其量化值。

解1. 由式 (2.18) 知直流分量的量化步长 $Q_{0,0} = 16$, 所以 100 的量化值为

$$\tilde{C}_{0,0} = \text{round} \left(\frac{C_{0,0}}{Q_{0,0}} \right) = \text{round} \left(\frac{100}{16} \right) = \text{round} (6.25) = 6 \quad (2.19)$$

可以通过调整量化值在低比特率和高图像质量之间取得折衷。量化系数的设计方法一方面源于人类知觉特征和心理视觉实验, 另一方面用到率失真理论。

2.3.2.4 熵编码

熵编码即对 DCT 系数 $C_{i,j}$ 的量化值 $\tilde{C}_{i,j}$ 进行无损编码, 其中对 DC 系数和 AC 系数采用不同的码本分开进行。

对 DC 系数先做差分编码 (Differential coding) 进行预处理 (类似于语音编码中的 DPCM)¹¹。首先将各个图像块的量化后的直流分量表示为一个矢量

$$\tilde{\mathbf{c}}_D = [\tilde{C}_{0,0}(1,1), \tilde{C}_{0,0}(1,2), \dots, \tilde{C}_{0,0}(1,W), \tilde{C}_{0,0}(2,1), \tilde{C}_{0,0}(2,2), \dots, \tilde{C}_{0,0}(H,W)], \quad (2.20)$$

¹¹现代通信原理课上讲解。

其中 H 和 W 分别表示横向和纵向的分块数。然后对上述信号进行差分运算，

$$\hat{c}_D(n) = \begin{cases} \tilde{c}_D(n) & n = 1; \\ \tilde{c}_D(n-1) - \tilde{c}_D(n) & \text{elsewhere} \end{cases} \quad (2.21)$$

得到变化更加平缓（想想为什么）的预测误差 \hat{c}_D ，接下来进行 Huffman 编码。每个预测误差 $\hat{c}_D(n)$ 可表示为 (Category, Magnitude) 的形式。Category 由预测误差的大小决定，用 Huffman 编码；Magnitude 即该误差的二进制表示（若为负数则表示为 1 的补码¹²）。直流分量预测误差的 Category 如表 2.2 所示¹³。注意直流分量预测误差的取值范围是 $[-2047, 2047]$ ，用 12bits 表示，分为 12 类。

表 2.2: 亮度直流分量预测误差的 Category 及其 Huffman 编码

预测误差	Category	Huffman 编码
0	0	00
-1, 1	1	010
-3, -2, 2, 3	2	011
-7, …, -4, 4, …, 7	3	100
-15, …, -8, 8, …, 15	4	101
-31, …, -16, 16, …, 31	5	110
-63, …, -32, 32, …, 63	6	1110
-127, …, -64, 64, …, 127	7	11110
-255, …, -128, 128, …, 255	8	111110
-511, …, -256, 256, …, 511	9	1111110
-1023, …, -512, 512, …, 1023	10	11111110
-2047, …, -1024, 1024, …, 2047	11	111111110

例 2.2. 已知某图像有三个块（好奇怪的图像），量化后的直流分量为 $\tilde{c}_D = [10, 8, 60]$ ，求其熵编码结果。

解 2. 首先进行差分编码，得到预测误差矢量为

$$\hat{c}_D = [10, 2, -52] \quad (2.22)$$

再逐一对三个预测误差进行编码：

1. 查表 2.2 可知，10 属于 Category 4，Huffman 码字是 101，10 的二进制表示是 1010，所以编码结果为 1011010；
2. 2 属于 Category 2，Huffman 码字是 011，2 的二进制表示是 10，所以编码结果为 01110；
3. -52 属于 Category 6，Huffman 码字是 1110，52 的二进制表示是 110100，则 -52 的 1 的补码为 001011，所以编码结果为 1110001011。

综上，题述直流分量的熵编码为 1011010011101110001011。

下面介绍 AC 系数的 Huffman 编码。JPEG 对每个块的量化 AC 系数依次进行熵编码。同 DC 系数，AC 系数在 Huffman 编码之前也要进行预处理，即 Zig-Zag 扫描和游程编码 (Run-length-coding)。Zig-Zag 扫描将二维的 DCT 系数矩阵变成一维矢量，扫描顺序如图 2.6 所示¹⁴。

¹²微机原理课上讲解。

¹³观察哪个区间对应的码字最短？这说明了什么？

¹⁴思考这么做的原因。

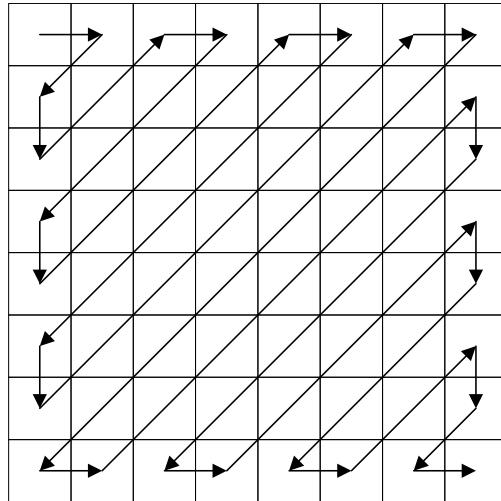


图 2.6: Zig-Zag 扫描示意图

游程编码按顺序依次输出每一个非零系数值及其行程 (Run)，即该系数前的零系数的个数¹⁵。从而每个非零系数可表示为 (Run/Size, Amplitude) 的形式，其中 Size 和 Amplitude 的意义分别和 DC 系数量化中的 Category 和 Magnitude 相同。即 AC 系数的 Size 由 Amplitude 确定，同表 2.2 左侧的两列的前 10 行¹⁶。和 DC 系数不同的是，熵编码对 AC 系数的 Run/Size ($0 \leq \text{Run} \leq 15$) 的联合体进行 Huffman 编码 (对应的，DC 系数只对 Category 做 Huffman 编码)，如表 2.3 所示。对 Amplitude 的编码方法和 DC 系数相同，仍以二进制 (正数) 或者 1 的二进制补码 (负数) 表示。注意 AC 系数编码过程将产生两个特殊符号：

- Zero Run Length (ZRL)：当 $\text{Run} > 15$ 时，即 $(\text{Run}/\text{Size}) = (15/0)$ ，插入符号 ZRL 表示 16 个连 0。
- End of Block (EOB)：编完最后一个非零的 AC 系数 (其后系数全为零)，插入块结束符 EOB，表 2.3 中记作 (0,0)。

例2.3. 求下面量化后 AC 系数块的熵编码。

$$\begin{bmatrix} 10 & 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.23)$$

解3. 1. Zig-Zag 扫描得到

$$10, 3, 0, 0, 2, \underbrace{0, \dots, 0}_{20 \text{ zeros}}, 1, \underbrace{0, \dots, 0}_{37 \text{ zeros}}, 0 \quad (2.24)$$

2. 第一个非零系数 10 的 Size 是 4，前面没有零所以 Run 是 0，查表得 (0/4) 的 Huffman 编码是 1011，再加上 10 的 Amplitude 1010，最终得到 10111010；

¹⁵插一句，黑白二值图像（如传真）的经典编码方法就是游程编码。

¹⁶因为量化后的 AC 系数取值范围为 $[-1023, 1023]$ ，因而分为 10 种 Size。

3. 对随后的非零系数 3 编码得到 0111；
4. 随后的非零系数是 2，它前面有 2 个零，查表 (2/2) 的编码是 11111000，还要加上 2 的 Amplitude 10；
5. 随后的非零系数是 1，它前面有 20 个零，先将 16 个连零 (F/0) 编码为 11111111001，再对 1 和它前面的 4 个零 (4/1) 编码为 111011，再加上 1 的 Amplitude 1；
6. 再无非零系数，对块结束 (EOB) 编码 1010；

综上，码流如下

10111010	0111	1111100010	111111110011110111	1010	(2.25)
10,	3,	0, 0, 2,	$\underbrace{0, \dots, 0}_{20 \text{ zeros}}, 1,$	$\underbrace{0, \dots, 0}_{37 \text{ zeros}}$	

下面总结 JPEG 中熵编码的特点：

- 分别处理 DC 和 AC 系数，因为二者的统计特性差别很大；
- 对式 (2.18) 中的典型量化步长，63 个 AC 系数中的很多都被量化为零。经 Zig-Zag 扫描后用 (Run/Size) 表示非零 AC 系数非常有效。
- DC 预测误差的取值范围是 [-2047, 2047]，AC 系数的范围是 [-1024, 1024]，若对这些值直接用 Huffman 编码分别需要 4095 项和 2047 项的码表。若只对 Size 或 (run/size) 编码，则码表尺寸分别减少为 12 项和 162 项¹⁷。

2.3.3 解码器

如图 2.7 所示，解码器和编码器的结构相似，工作流程相反。解码器主要包括熵解码、反量化、离散余弦逆变换 (IDCT) 等几部分。

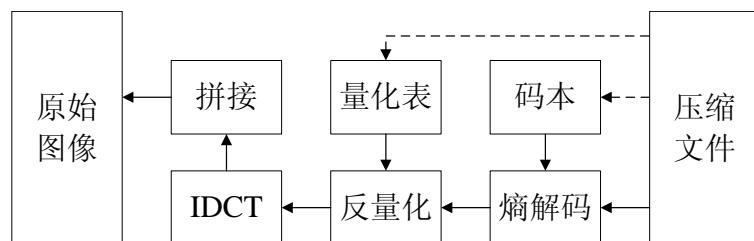


图 2.7: 解码器结构框图

2.3.3.1 熵解码

Huffman 码是前缀码，即任何码字都不是其他码字的前缀，所以即使码字间没有分隔符也可无误的解码。

对 DC 系数，首先用 Huffman 解码解出 Category，再查表得到对应 Magnitude 的长度，进而取出这些 bits。若首 bit 是 1，则将二进制直接表示成十进制即可；若是 0 则表示 1 的补码，对应的 Magnitude 是负

¹⁷这一条解释起来有点复杂，感兴趣的同学可选修图像编码方面的课程。

数，则先逐 bit 取反，再译成十进制后加个负号。依次用上述步骤即可解出所有块的量化后 DC 系数的预测误差，再参考式 (2.21) 恢复所有量化后的 DC 系数即可。

对 AC 系数，首先用 Huffman 解码解出 (Run/Size)，再查表得到对应 Amplitude 的长度，同解 DC 系数的方法解出一个非零系数（和前面 Run 个零），进而依次解码直到 EOB 后用零补足 63 个系数为止。这样就解出了一个块的 AC 系数。同理依次解出所有块的系数。

2.3.3.2 反量化

反量化是量化的逆过程，即用量化后的系数乘以量化步长恢复原系数

$$\bar{C}_{i,j} = \tilde{C}_{i,j} Q_{i,j}, \quad \forall i, j = 0, 1, \dots, N - 1 \quad (2.26)$$

明显复原的系数和真实系数有误差¹⁸。此误差直接导致解码恢复后的图像和原图像不同。

2.3.3.3 DCT 逆变换

参见式 (2.16) 和 (2.17)，不再赘述。

2.3.3.4 拼接

不要忘记拼接时在每个像素值上加上 128。

2.3.4 码流组织

JPEG 定义了一个用于图像文件编解码的交换格式，但这种格式居然没定义所使用的色彩空间。所以后来又出现了一种称为 JPEG 文件交换格式 (JPEG file interchange format, JFIF) 的标准。JFIF 是 JPEG 的超集，它直接使用前者为应用程序定义的许多标记，因此 JFIF 格式成了事实上标准的 JPEG 文件交换格式。现有的 JPEG 文件大多基于此标准组织。

JPEG 的每个标记都由 2 个字节组成，其前一个字节是固定值 0xFF。每个标记之前还可以添加数目不限的 0xFF 填充字节。表 2.4 列出了其中的 8 个标记¹⁹。

2.3.5 补充知识

作为国际标准化组织定义的标准，JPEG 代表了当时图像编码学术界的最高水平，它具有计算量适中²⁰，支持在压缩比和图像质量间选择折衷，同时不依赖于图像类型，适用于各种图像源、图像内容、彩色空间、图像尺寸和像素分辨率等优点。

本章介绍的流程仅仅是 JPEG 编码的一个子集。作为一种通用的静止图像编码方法，JPEG 可适应各种不同的压缩要求，比如既可单扫描顺序 (sequential) 编码，也允许多次扫描的累进编码 (progressive)，同时还提供分层编码的选择，低分辨率已满足时可不必解高分辨率图像。

最后我们介绍一种编码后图像质量的评价方式。峰值信噪比 (Peak Signal to Noise Ratio, PSNR) 是最广泛使用的评价图像质量的客观标准。它是原图像与处理图像之间均方误差相对于 $(2^n - 1)^2$ 的对数值（即信号最大值的平方，其中 n 是每个采样值的比特数）。本章中对应的定义为

$$\text{PSNR} = 10 \lg \left(\frac{255^2}{\text{MSE}} \right) \quad (2.27)$$

¹⁸误差的范围取决于什么？

¹⁹同学们可以尝试用 UltraEdit 等二进制文件查看软件打开一个 JPEG 文件，找找其中是否有上述标记。JFIF 不涉及更多学术性的知识，只是一个格式罢了，所以不再过多讲解。

²⁰新标准 JPEG2000 比 JPEG 压缩率更高，功能更丰富，但过大的计算量是影响其普及的重要因素。

其中 255 就是 8 bits 表示法的最大值 (Peak) , 而 MSE 表示原图像与处理图像之间均方误差 (Mean Square Error)

$$\text{MSE} = \frac{1}{\text{PixelNum}} \sum_x \sum_y (P_{x,y} - \hat{P}_{x,y})^2 \quad (2.28)$$

PSNR 的单位为 dB 。所以 PSNR 值越大表示失真越小。

第四节 练习题

本章练习题所用数据均可由 “ JpegCoeff.mat ” 导入，其内容如表 2.5 所示。本章练习题中 “测试图像” 指的是 hall.mat 中的灰度图像。

1. 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。
2. 请编程实现二维 DCT , 并和 MATLAB 自带的库函数 dct2 比较是否一致。
3. 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？
4. 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。
5. 如果认为差分编码是一个系统，请绘出这个系统的频率响应，说明它是一个_____（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。²¹
6. 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 ？
7. 你知道哪些实现 Zig-Zag 扫描的方法？请利用 MATLAB 的强大功能设计一种最佳方法。
8. 对测试图像分块、DCT 和量化，将量化后的系数写成矩阵的形式，其中每一列为一个块的 DCT 系数 Zig-Zag 扫描后形成的列矢量，第一行为各个块的 DC 系数。
9. 请实现本章介绍的 JPEG 编码（不包括写 JFIF 文件），输出为 DC 系数的码流、 AC 系数的码流、图像高度和图像宽度，将这四个变量写入 jpegcodes.mat 文件。
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。
11. 请实现本章介绍的 JPEG 解码，输入是你生成的 jpegcodes.mat 文件。分别用客观 (PSNR) 和主观方式评价编解码效果如何。
12. 将量化步长减小为原来的一半，重做编解码。同标准量化步长的情况比较压缩比和图像质量。
13. 看电视时偶尔能看到美丽的雪花图像（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。

²¹希望同学们现在对压缩编码能有些深层次理解，不理解也没关系，信息论课上会讲。

表 2.3: 亮度 AC 分量的 Run/Size 及其 Huffman 编码 (部分)

Run/Size	码长	码字	Run/Size	码长	码字
0/0 (EOB)	4	1010			
0/1	2	00	4/1	6	111011
0/2	2	01	4/2	10	1111111000
0/3	3	100	4/3	16	1111111110010111
0/4	4	1011	4/4	16	1111111110011000
0/5	5	11010	4/5	16	1111111110011001
0/6	6	111000	4/6	16	1111111110011010
0/7	7	1111000	4/7	16	1111111110011011
0/8	10	1111110110	4/8	16	1111111110011100
0/9	16	1111111110000010	4/9	16	1111111110011101
0/A	16	1111111110000011	4/A	16	1111111110011110
1/1	4	1100	8/1	8	11111010
1/2	6	111001	8/2	15	1111111110000000
1/3	7	1111001	8/3	16	1111111110110111
1/4	9	111110110	8/4	16	1111111110111000
1/5	11	11111110110	8/5	16	1111111110111001
1/6	16	1111111110000100	8/6	16	1111111110111010
1/7	16	1111111110000101	8/7	16	1111111110111011
1/8	16	1111111110000110	8/8	16	1111111110111100
1/9	16	1111111110000111	8/9	16	1111111110111101
1/A	16	1111111110001000	8/A	16	1111111110111110
			F/0 (ZRL)	11	11111111001
2/1	5	11011	F/1	16	111111111110101
2/2	8	11111000	F/2	16	1111111111110110
2/3	10	1111110111	F/3	16	11111111111110111
2/4	16	1111111110001001	F/4	16	11111111111111000
2/5	16	1111111110001010	F/5	16	11111111111111001
2/6	16	1111111110001011	F/6	16	11111111111111010
2/7	16	1111111110001100	F/7	16	11111111111111011
2/8	16	1111111110001101	F/8	16	11111111111111100
2/9	16	1111111110001110	F/9	16	111111111111111101
2/A	16	1111111110001111	F/A	16	111111111111111110

表 2.4: JFIF 主要标记

序号	符号	标记	说明
1	SOI	0xD8	图像开始
2	APP0	0xE0	JFIF 应用数据块
3	APPn	0xE1 - 0xEF	其他的应用数据块(n, 1~15)
4	DQT	0xDB	量化表
5	SOF0	0xC0	帧开始
6	DHT	0xC4	Huffman 表
7	SOS	0xDA	扫描线开始
8	EOI	0xD9	图像结束

表 2.5: JpegCoeff.mat 中所含数据

变量名	含义	说明
QTAB	DCT 系数的量化步长矩阵, 式 (2.18)	
DCTAB	DC 系数预测误差的 Category 的码本, 表 2.2	每行对应一个 Catrgory , 第一列对应 Huffman 编码的长度 L , 随后 L 列对应该码字, 再后全零为填充物
ACTAB	AC 系数的 (Run/Size) 的码本, 完整的表 2.3	每行对应一个 (Run/Size) , 第一列表示 Run , 第二列表示 Size , 第三列表示该 (Run/Size) 对应 Huffman 编码的长度 L , 随后 L 列对应该码字, 再后全零为填充物

第三章 信息隐藏

基于上一章变换域图像编码的基础，本章介绍一种在变换域进行信息隐藏的方法。

第一节 背景知识

隐写术是一种古老的技术，即将特定信息秘密的嵌入某种常见的载体上，通过传递该载体达到保密通信的目的，比如一幅普通的画卷内可能藏有珍宝图或者进攻路线图。又如我国历代文人墨客喜欢卖弄的藏头诗，或者传说中二战期间美国西海岸某海港城市童心未泯的老太太每周寄往日本的布娃娃，据说她用布娃娃的个数和大小暗示离港军舰的数量和型号。

信息隐藏可以认为是广义的隐写术，它和加密有本质的区别

- 你能轻易看出某个文件被加过密，甚至你也知道加密方法的详细步骤，但因为不知道密码而无法（没时间）解密成明文。
- 信息隐藏的输出表现为一个很正常的文件，比如一个风景图片或者一个说明文档，但其中却隐藏了特定信息，先不说提取信息的难度如何，单是判断是否隐藏了信息就是一个很难的任务。

数字水印是一种应用广泛的信息隐藏技术：制造商为了保护版权，在其生产的媒体产品中嵌入特定的标志，一方面该标志不会影响消费者使用该媒体，另一方面也不会轻易被盗版者攻击或破解。

以图像为载体的信息隐藏流程如图 3.1 所示。

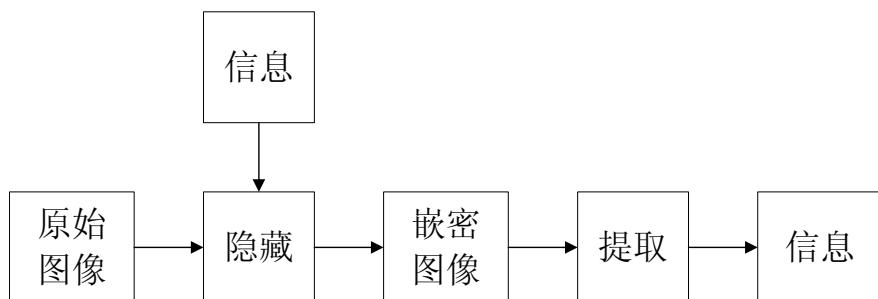


图 3.1: 信息隐藏和提取流程图

第二节 空域信息隐藏技术

空域信息隐藏就是直接将信息数据插到像素的亮度或者色度值中。比如先将信息表示成二进制码流，再

依次用每位信息替换掉图像中各像素亮度分量的最低位。

这样做的优点是简单且数据隐藏量很大—8个bits就可以隐藏1个bit。缺点是只能以未压缩的格式存储图像，因为图像一经有损压缩，附着信息会被当作是高频分量舍弃掉。而在JPEG压缩格式充斥网络的今天，发送一个BMP文件就显得太突兀了，有违安全隐蔽的初衷。

第三节 DCT 域信息隐藏技术

在刚刚学过JPEG图像压缩编码的基础上，很容易想到将信息隐藏在DCT域，隐藏方法有多种。

1. 同空域方法，用信息位逐一替换掉每个量化后的DCT系数的最低位，再行熵编码。
2. 同方法1，用信息位逐一替换掉若干量化后的DCT系数的最低位，再行熵编码。注意不是每个DCT系数都嵌入了信息。
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；

无论上述哪种方法，信息隐藏步骤都位于量化之后，熵编码之前，如图3.2所示。

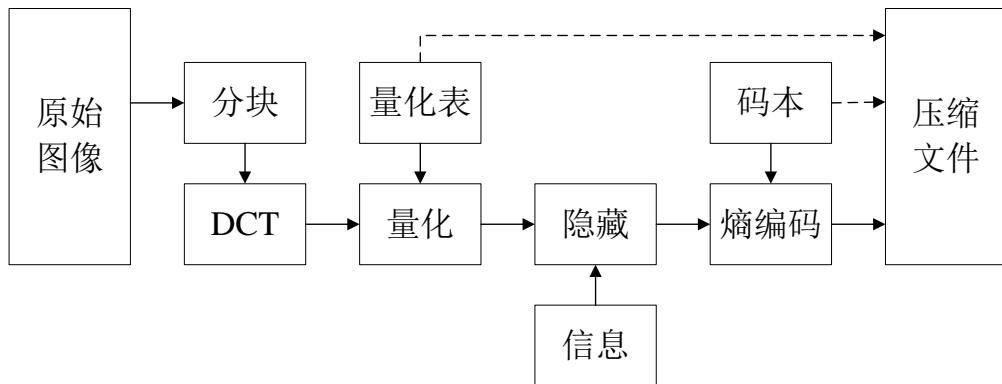


图 3.2: DCT 域信息隐藏结构图

第四节 练习题

本章练习题所用测试图像同上一章。本章练习题所指待隐藏信息可自由选择。

1. 实现本章介绍的空域隐藏方法和提取方法。验证其抗JPEG编码能力。
2. 依次实现本章介绍的三种变换域信息隐藏方法和提取方法，分析嵌密方法的隐蔽性以及嵌密后JPEG图像的质量变化和压缩比变化。
3. (选做) 请设计实现新的隐藏算法并分析其优缺点。

第四章 人脸检测

作为图像处理大作业的最后一部分，本章希望延续郑版《信号与系统》具有时代气息的优良传统。

第一节 背景知识

人脸检测与识别是模式识别领域的重要问题，也是智能视频监测和安防应用所需的关键技术。人脸检测即在复杂背景的图像中确定人脸的位置，它可以看成是一个特殊的两类识别问题，即寻找所有人脸的共同特征来区分所有非人脸的特征，而人脸识别是要寻找每张人脸的特征以区别于不同的人脸。

目前人脸检测的经典方法是 Adaboost，人脸识别的经典方法是基于主成份分析 (PCA) 的特征脸 (Eigenface) 技术及其各种衍生算法。鉴于同学们目前知识结构的局限，我们在这里不展开介绍这些方法，有兴趣的同学可以通过选修相关课程、阅读参考文献和自己动手实践来学习和掌握这些方法。

本章将介绍一种基于肤色的简单检测方法。

第二节 基于彩色直方图的人脸检测方法

人脸检测系统的结构如图 4.1 所示。图中虚线框内表示训练过程，即由人脸样本提取标准特征的过程。训练必须先于检测完成，训练所得标准特征将存储起来供检测使用。

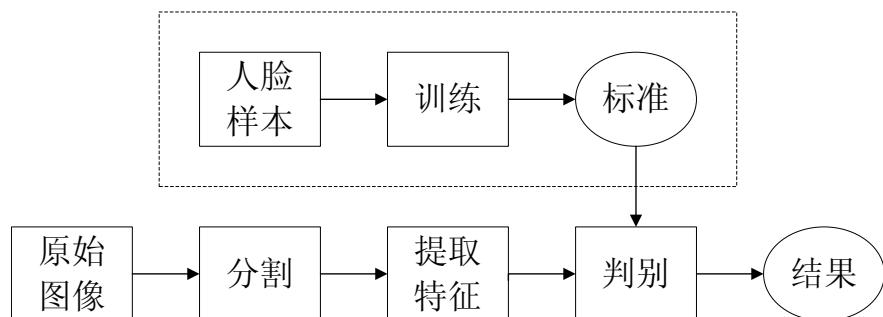


图 4.1: 人脸检测结构框图

4.2.1 选取特征

我们希望以颜色作为人脸图像的特征，即用各种不同颜色的比例构成的矢量作为确定人脸的标准。换

换句话说，对任意给定的一块区域，我们用各种颜色出现的频率作为该区域的特征，即

$$\mathbf{u}(R) = [f_0(R), f_1(R), \dots, f_{N-1}(R)]^T \quad (4.1)$$

其中 N 表示颜色的总数（以 3 字节 RGB 表示的色彩空间—即所谓的 24 位真彩色—为例，总共有 $N = 2^{3 \times 8} = 16777216$ 种颜色）， $f_n(R)$ 表示第 n 种颜色在区域 R 内出现的频率，即表现为第 n 种颜色的像素点数占区域内总像素点数的比例，

$$f_n(R) = \frac{\sum_{(x,y) \in R} \delta(P_{x,y} - c(n))}{\sum_{(x,y) \in R} 1} \quad (4.2)$$

其中 $P_{x,y}$ 和 $c(n)$ 分别表示位于 (x, y) 的像素的颜色和第 n 种颜色，它们都是三元组，即

$$P_{x,y} = [R_{x,y}, G_{x,y}, B_{x,y}] \quad (4.3)$$

$$c(n) = [n^{(24 \sim 17)}, n^{(16 \sim 9)}, n^{(8 \sim 1)}] \quad (4.4)$$

其中 $n^{(i \sim j)}$ 的含义是先将 n 表示成 2 进制数，再取其第 i 到第 j 个 bits 组成一个新的数并译回 10 进制。假设 n 是一个用 3 个字节表示的数，则 $n^{(24 \sim 17)}$ 表示其最高字节代表的数，同理， $n^{(16 \sim 9)}$ 和 $n^{(8 \sim 1)}$ 分别代表其中间字节和最低字节代表的数。换句话说，我们有

$$n = (n^{(24 \sim 17)} << 16) + (n^{(16 \sim 9)} << 8) + n^{(8 \sim 1)} \quad (4.5)$$

这样做的目的是在自然数 n 和用 RGB 三元组表示的颜色之间建立一种简单的一一对应关系。

现在回过头来看 (4.2) 中的 $\delta(\cdot)$ ，它可以定义为

$$\delta(P_{x,y} - c(n)) = \begin{cases} 1 & R_{x,y} = n^{(24 \sim 17)}, G_{x,y} = n^{(16 \sim 9)}, B_{x,y} = n^{(8 \sim 1)} \\ 0 & \text{elsewhere} \end{cases} \quad (4.6)$$

也可以定义为

$$\delta(P_{x,y} - c(n)) = \begin{cases} 1 & (R_{x,y} << 16) + (G_{x,y} << 8) + B_{x,y} = n \\ 0 & \text{elsewhere} \end{cases} \quad (4.7)$$

明显这两种定义完全相同。

很容易发现式 (4.1) 中的定义的特征实际上是一个概率密度函数，即满足

$$\sum_{n=0}^{N-1} f_n(R) = 1, \quad \forall R \quad (4.8)$$

这样我们就定义了一种利用颜色提取图像特征的方法。它很简单，而且很有意义。但这个特征有个明显的缺点就是数据量太大了——无论待处理的图有多么小，这个特征都需要 N 个浮点数的存储单元。过高维的特征带来的另一个问题是增加了检测—即判定两个特征之间相似程度—的计算量。它的最后一个缺点是稳健性差，试想：光照稍微变化，相片上的色彩也会变化，任一个彩色分量哪怕只是增加或减少了 1，也将导致 $\mathbf{u}(R)$ 发生改变。这种对光照的敏感性极大得降低了上述特征的实用性。

更经济些也更鲁棒些的方法是减小颜色的种类，与其用 $N = 2^{3 \times 8}$ 种颜色，还不如用 $M = 2^{3 \times L}$ 种，其中 $L = 3$ 或者任意小于 8 的值。这样一来颜色的数量减小了，同时 $\mathbf{u}(R)$ 的长度也减小了。请同学们思考此时的式 (4.4) 和 (4.7) 应如何定义？

以颜色做人脸特征的优点是计算量小和抗旋转，缺点是容易对皮肤和其他肤色相近的物体产生误判¹，而且对相片的光照强度和方向、阴影等比较敏感。另外，以黄色人种训练出的标准检测其它人种准确度会大大下降，当然对彩绘等装饰也非常敏感。

另一种典型特征是边缘或轮廓，即以脸庞和五官等纹理作为特征。这样做的优点是对光照、色彩非常鲁棒，缺点是算法复杂，要求人脸必须正面且不能大角度旋转。

¹藉此可解释机器猫通过加菲猫却通不过的原因。

4.2.2 训练标准特征

训练方法非常简单，就是收集尽可能多的人脸相片，比如 I 张，逐一用手工准确定位第 i 张图内人脸的位置 R_i ，然后计算该区域内的特征 $\mathbf{u}(R_i)$ ，最后统计其平均值 \mathbf{v}

$$\mathbf{v} = \frac{1}{I} \sum_i \mathbf{u}(R_i) \quad (4.9)$$

矢量 \mathbf{v} 即是训练出的人脸标准。

4.2.3 检测

检测即判断某一区域的特征和人脸标准的相近程度，如果小于阈值则判为人脸

$$R \in \text{Face} \quad \text{if } d(\mathbf{u}(R), \mathbf{v}) < \epsilon \quad (4.10)$$

其中 $d(\cdot)$ 表示距离函数， ϵ 表示阈值。

对于分别满足 $\sum u_n = 1$ 和 $\sum v_n = 1$ 的两个矢量 $\mathbf{u}(R)$ 和 \mathbf{v} ，Bhattacharyya 提出了一种度量方法

$$\rho(\mathbf{u}, \mathbf{v}) = \sum_n \sqrt{u_n v_n} = \cos \theta \quad (4.11)$$

显而易见， θ 表示 N 维空间中两个矢量 $[\sqrt{u_0}, \sqrt{u_1}, \dots, \sqrt{u_{N-1}}]^T$ 和 $[\sqrt{v_0}, \sqrt{v_1}, \dots, \sqrt{v_{N-1}}]^T$ 的夹角，当两个矢量完全相同时，

$$\rho = \cos 0 = 1 \quad (4.12)$$

所以待定特征和人脸标准的距离可定义为

$$d(\mathbf{u}(R), \mathbf{v}) = 1 - \rho(\mathbf{u}(R), \mathbf{v}) = 1 - \sum_n \sqrt{u_n v_n} \quad (4.13)$$

第三节 练习题

1. 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 \mathbf{v} 。
 - (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？
 - (b) 假设 L 分别取 3, 4, 5，所得三个 \mathbf{v} 之间有何关系？
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（输出图像在判定为人脸的位置加上红色的方框）。随意选取一张多人照片（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。
3. 对上述图像分别进行如下处理后
 - (a) 顺时针旋转 90° (imrotate)；
 - (b) 保持高度不变，宽度拉伸为原来的 2 倍 (imresize)；
 - (c) 适当改变颜色 (imadjust)；
 再试试你的算法检测结果如何？并分析所得结果。
4. 如果可以重新选择人脸样本训练标准，你觉得应该如何选取？