信息论 project
JPEG2000

高云帆
张奕朗 16307130242
陈幸豪 16307130006

2018 年 12 月 29 日

1 综述

高云帆：

张奕朗：

陈幸豪：色彩空间变换，量化，搭整个程序的框架，设计协调整个流程的数据结构.

2 图像分块与电平平移归一化

在图像变换之前，首先需要对图像进行预处理. 预处理主要分为分块、电平平移归一化和颜色变换三个过程（**图2.1**）. 在调用 *cv2.imread* 读取图像数据时，利用正则表达式获取图像的颜色位数信息，方便之后的处理. 代码见函数 *init_image*.

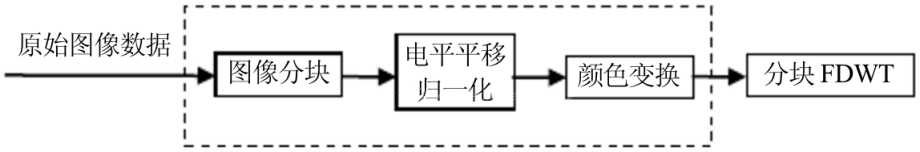

图 2.1 图像预处理.

2.1 图像分块

与 JPEG 标准相同，JPEG2000 标准也需要将图像分块，不过是为了降低计算量和对内存的需求，以方便压缩（当然也可以选择不分块）. 但与 JPEG 标准不同的是，JPEG2000 标准并不需要将图像强制分成 8×8 的小块, 而是可以将图像分割成若干任意大小的互不重叠的矩形块 (tile), 常分为 $2^6 \sim 2^{12}$（即 64 ~ 1024 像素宽）的正方形 tile. 受图像形状的影响，边缘部分的 tile 的大小和形状可能与其它的 tile 不同. Tile 的大小会影响重构图像的质量. 一般情况下，大的 tile 重构出的图像质量要好一些，因为小波变换的范围大，减轻了边缘效应. 代码见 *class Tile* 和 *class JPEG2000* 中的函数 *image_tiling*.

2.2 电平平移归一化

电平平移归一化分为两步：

第一步是直流电平平移（DC level shifting），通过将数据减去均值使之关于 0 对称分布，以去掉频谱中的直流分量.

$$I_1(x,y) = I(x,y) - 2^{B-1}$$
$$-2^{B-1} \le I_1(x,y) \le 2^{B-1} - 1$$

其中 B 是之前读取到的颜色位数信息.

第二步是电平归一化（normaliztion）. 对于无损压缩中的 5/3 小波变换，由于采用的是整数小波变换，所以不需要进行归一化；而对于有损压缩中的 9/7 小波变换，由于采用的是实数小波变换，故需要对每个 tile 进行如下运算以归一化：

$$I_2(x,y) = \frac{I_1(x,y)}{2^B}$$
$$-\frac{1}{2} \le I_2(x,y) < \frac{1}{2}$$

但是事实上，电平平移归一化对 DWT 来说不是必须的（而且电平平移归一化后还不便于进行色彩空间变换），因为它只会影响小波系数的动态范围而不会影响结果. 代码见函数 *dc_level_shift*.

2.3 反变换

在解码、小波反变换和色彩空间反变换之后，需要进行电平反平移，（如果是有损压缩，还需要进行反归一化），最后拼接得到恢复的图像. 这些都是上述操作的反过程，这里不再赘述. 代码见函数 *idc_level_shift* 和 *image_splicing*.

3 颜色空间变换

通常所谓的颜色由红绿蓝（RGB）组成，即 RGB 形成了颜色空间的一组基. 其他常用颜色空间有 CMY，HSV，YCbCr 等. JPEG2000 标准中，可以选择将 RGB 颜色变换到其他空间. 需要注意的是，图像可能有超过 3 个颜色分量，而 JPEG2000 的颜色变换只运用于前 3 个颜色分量. 此外，变换过程在理论上不区分输入的颜色空间，即输入颜色不一定须为 RGB 数据. 颜色空间变换的目的是利用三个图像分量组合之间的冗余. 例如人眼对 YCbCr 空间的 Y 分量更敏感，而对 Cb、Cr 分量相对不敏感. 此时可对 Y 分量做无损压缩，而对 CbCr 做有损压缩. 此时图片的体积可被减小，而观感相比无损压缩没有明显损失.

JPEG2000 定义了不可逆彩色变换（ICT）和可逆彩色变换（RCT）. ICT 计算过程中使用的小数难以被二进制精确表示，即必然存在计算误差. 因此无损压缩必须使用 RCT. 代码中颜色空间变换的函数名为 *component_transformation*，可按压缩方式选择 ICT 或 RCT. 下面分别介绍两种变换.

3.1 ICT

ICT 可视为一个线性变换. 它将归一化到 $\left[-\frac{1}{2},\frac{1}{2}\right]$ 的 RGB 彩色变换到 YCbCr 空间. 首先定义以下加权因子

$$\alpha_R = 0.299, \alpha_G = 0.587, \alpha_B = 0.114 \tag{1}$$

以及变换关系

$$\begin{aligned} x_Y[n] &= \alpha_R x_R[n] + \alpha_G x_G[n] + \alpha_B x_B[n] \\ x_{Cb}[n] &= \frac{0.5}{1-\alpha_B}(x_B[n] - x_Y[n]) \\ x_{Cr}[n] &= \frac{0.5}{1-\alpha_B}(x_R[n] - x_Y[n]) \end{aligned} \tag{2}$$

这一关系可用矩阵表示如下

$$\begin{pmatrix} x_Y[n] \\ x_{Cb}[n] \\ x_{Cr}[n] \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{pmatrix} \begin{pmatrix} x_R[n] \\ x_G[n] \\ x_B[n] \end{pmatrix} \tag{3}$$

需要注意，以上矩阵中使用的小数都是近似值. 实际计算中，所有数值必须精确处理.

在有损压缩中，若输入的 RGB 值域已被归一化到 $\left[-\frac{1}{2},\frac{1}{2}\right]$，则不难证明，输出的 YCbCr 值域也在 $\left[-\frac{1}{2},\frac{1}{2}\right]$. 其中 x_Y 是 RGB 分量的加权平均，被认为是图像强度（或亮度）的量度. 该权重了反映可见光谱中绿色段信息对于视觉感知的重要性.

解压缩过程中，对应的 ICT 反变换如下

$$\begin{pmatrix} x_R[n] \\ x_G[n] \\ x_B[n] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.402 \\ 1 & -0.344136 & -0.714136 \\ 1 & 1.772 & 0 \end{pmatrix} \begin{pmatrix} x_Y[n] \\ x_{Cb}[n] \\ x_{Cr}[n] \end{pmatrix} \tag{4}$$

其中仅第 1 行和第 3 行的小数为精确值.

ICT 反变换不具有正向变换的范围保持特性. 此外，由于输入的 $x_Y[n]$、$x_{Cb}[n]$ 和 $x_{Cr}[n]$ 通常已被量化所损坏，所以由 ICT 反变换重建的 RGB 值不一定落在 $\left[-\frac{1}{2}, \frac{1}{2}\right]$. 解压缩程序可对超出范围的 RGB 样本值进行处理，但 JPEG2000 标准并未指定处理方法.

3.2 RCT

RCT 将值域为 $\left[-2^B, 2^B - 1\right]$ 的 RGB 彩色变换到 Y'DbDr 空间. RCT 由以下非线性关系定义

$$\begin{aligned} x'_Y[n] &= \left\lfloor \frac{x_R[n] + 2x_G[n] + x_B[n]}{4} \right\rfloor \\ x_{Db}[n] &= x_B[n] - x_G[n] \\ x_{Dr}[n] &= x_R[n] - x_G[n] \end{aligned} \tag{5}$$

其中 $\lfloor x \rfloor$ 表示小于等于 x 的最大整数.

可以证明 RCT 能被精确地反变换. 由于

$$\begin{aligned} \left\lfloor \frac{x_{Db}[n] + x_{Dr}[n]}{4} \right\rfloor &= \left\lfloor \frac{x_R[n] + 2x_G[n] + x_B[n]}{4} - x_G[n] \right\rfloor \\ &= x'_Y[n] - x_G[n] \end{aligned} \tag{6}$$

因此我们可以先重建 $x_G[n]$，再借此重建 $x_R[n]$ 和 $x_B[n]$. 定义 RCT 反变换为

$$\begin{aligned} x_G[n] &= x'_Y[n] - \left\lfloor \frac{x_{Db}[n] + x_{Dr}[n]}{4} \right\rfloor \\ x_B[n] &= x_{Db}[n] + x_G[n] \\ x_R[n] &= x_{Dr}[n] + x_G[n] \end{aligned} \tag{7}$$

RCT 也可以近似取一种变换关系如下

$$\begin{pmatrix} x'_Y[n] \\ x_{Db}[n] \\ x_{Dr}[n] \end{pmatrix} = \begin{pmatrix} 0.25 & 0.5 & 0.25 \\ 0 & -1 & 1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_R[n] \\ x_G[n] \\ x_B[n] \end{pmatrix} \tag{8}$$

其反变换可近似为

$$\begin{pmatrix} x_R[n] \\ x_G[n] \\ x_B[n] \end{pmatrix} = \begin{pmatrix} 1 & -0.25 & -0.25 \\ 1 & -0.25 & 0.75 \\ 1 & 0.75 & -0.25 \end{pmatrix} \begin{pmatrix} x'_Y[n] \\ x_{Db}[n] \\ x_{Dr}[n] \end{pmatrix} \tag{9}$$

这里的近似仅仅修改了变换关系，没有对任何数值进行舍入处理. 所以近似变换依然是无损的.

4 小波变换

4.1 小波变换原理简述

对非平稳过程，傅里叶变换有其局限性. 它只能获取一段信号总体上包含哪些频率的成分，但是对各成分出现的时刻并无所知. 因此时域相差很大的两个信号，可能频谱图一样. 而加窗截取信号的 STFT（短时傅里叶变换）又存在窗长的问题：窗太窄，频率分辨率差；窗太宽，时间分辨率低. 于是便产生了利用有限长的会衰减的小波基进行与源信号进行相关的 WT（小波变换）和 DWT（离散小波变换），需要用到的 DWT 的变换和逆变换公式如下：

$$WT[j,k] = \sum_n \langle f, \psi_{j,k} \rangle = \frac{1}{\sqrt{j}} \sum_n f[n] \psi[\frac{n-k}{j}]$$

$$f[n] = \sum_{j,k} \langle f, \psi_{j,k} \rangle \hat{\psi}_{j,k} = \frac{1}{A} \sum_{j,k} WT[j,k] \psi_{j,k}$$

不同于傅里叶变换，变量只有频率 ω，小波变换有两个变量：尺度 j 和平移量 k. 尺度 j 控制小波函数的伸缩，平移量 k 控制小波函数的平移. 尺度 j 对应于频率（反比），平移量 k 对应于时间.

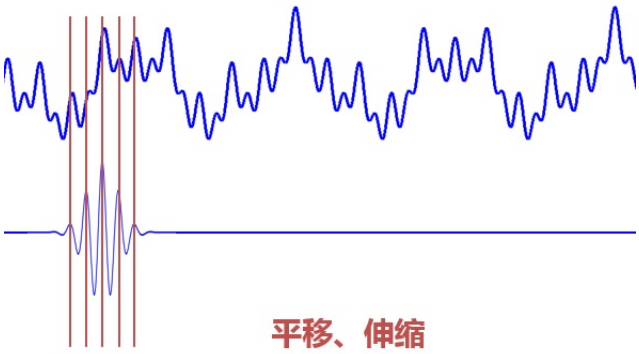

图 4.2 小波基的尺度参数和平移参数.

4.2 小波变换的滤波器实现与提升算法

小波变换的滤波器等效

小波变换可以用 IIR 滤波器进行等效（具体推导见《JPEG2000 图像压缩基础、标准和实践》），并且可以通过提升算法进行实现. 在有损压缩的情况下，JPEG2000 标准的核心编码系统默认的不可逆小波变换是 Daubechies 9/7 DWT 的提升实现，而在无损情况下则采用 Le Gall 5/3 滤波器的提升实现的整数可逆小波变换. 其中 Daubechies 9/7 是 I.Daubechies 与 M.Antonini 等人于 1992 年提出的一种双正交小波滤波器. Le Gall 5/3 是 D.Le Gall 与 A.Tabatabai 于 1988 年基于样条 5/3 变换而提出的一种可逆双正交滤波器. 两者的滤波器参数见**图4.3**、**4.4**. 其中参数 9/7 和 5/3 分别指分解低/高通滤波器的 IIR 长度，合成滤波器相反（正交对称）.

	分解滤波器系数		合成滤波器系数	
	低通滤波器 $h_L(i)$	高通滤波器 $h_H(i)$	低通滤波器 $g_L(i)$	高通滤波器 $g_H(i)$
0	0.6029490182363579	1.115087052456994	1.115087052456994	0.6029490182363579
±1	0.2668641184428723	-0.5912717631142470	0.5912717631142470	-0.2668641184428723
±2	-0.07822326652898785	-0.05754352622849957	-0.05754352622849957	-0.07822326652898785
±3	-0.01686411844287495	0.09127176311424948	-0.09127176311424948	0.01686411844287495
±4	0.02674875741080976	0	0	0.02674875741080976
其他值	0	0	0	0

图 4.3　Daubechies 9/7 滤波器系数表.

i	分解滤波器系数		合成滤波器系数	
	低通滤波器 $h_L(i)$	高通滤波器 $h_H(i)$	低通滤波器 $g_L(i)$	高通滤波器 $g_H(i)$
0	6/8	1	1	6/8
±1	2/8	-1/2	1/2	-2/8
±2	-1/8	0	0	-1/8
其他值	0	0	0	0

图 4.4　Le Gall 5/3 滤波器系数表.

小波变换的提升算法

Daubechies 9/7　正变换包括 4 步提升和 2 步缩放：

4 步提升：

$$y(2n+1) = x(2n+1) - \alpha[x(2n) + x(2n+2)]$$

$$y(2n) = x(2n) - \beta[y(2n-1) + y(2n+1)]$$

$$y(2n+1) = y(2n+1) + \gamma[y(2n) + y(2n+2)]$$

$$y(2n) = y(2n) + \delta[y(2n-1) + y(2n+1)]$$

2 步缩放：

$$y(2n+1) = -K\,y(2n+1)$$

$$y(2n) = y(2n)/K$$

逆变换包括 2 步缩放和 4 步提升：

2 步缩放：

$$x(2n+1) = -y(2n+1)/K$$

$$x(2n) = K\,y(2n)$$

4 步提升：

$$x(2n) = x(2n) - \delta * [x(2n-1) + x(2n+1)]$$

$$x(2n+1) = x(2n+1) - \gamma * [x(2n) + x(2n+2)]$$

$$x(2n) = x(2n) + \beta * [y = x(2n-1) + x(2n+1)]$$

$$x(2n+1) = x(2n+1) + \alpha * [x(2n) + x(2n+2)]$$

其中 $\alpha, \beta, \gamma, \delta, K$ 为与滤波器系数有关的参数：$\alpha = 1.586134342, \beta = 0.052980118, \gamma = 0.882911075, \delta = 0.443506852, K = 1.230174105$.

Le Gall 5/3 正、逆变换均只需要一步提升与一步缩放.

正变换:

$$y(2n+1) = x(2n+1) - [x(2n) + x(2n+2)]/2$$
$$y(2n) = x(2n) + [y(2n-1) + y(2n+1) + 2]/4$$

逆变换:

$$x(2n) = y(2n) - [y(2n-1) + y(2n+1) + 2]/4$$
$$x(2n+1) = y(2n+1) + [x(2n) + x(2n+2)]/2$$

4.3 二维多级小波变换

对图像进行 2D-DWT 后可得到图像的 2D-DWT 系数，将系数按照左上、左下、右上、右下分别分为 LL、LH、HL、HH 四个分量（L 表示低频，H 表示高频）. 由于图像的信息大多集中在低频区域，所以若分解所得的 LL 分量仍含有较多的图像信息时（动态范围大，不利于量化和编码），可将其进一步做 2D-DWT 分解，也就是多级小波分解，如**图4.5**所示. JPEG2000 中一般做 3 ~ 4 级 2D-DWT，经实践，选用 3 级 2D-DWT 就已经足够了.

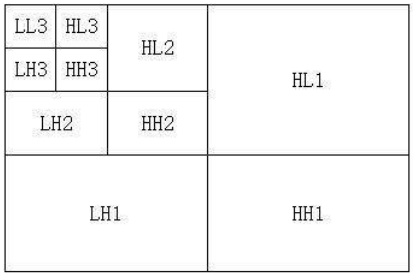

图 4.5 二维小波变换的多级划分.

开始时我用 *for* 循环实现了提升算法，但是运行速度太慢，于是改为调用 *python* 的 PyWavelets 包，建立自定义小波来实现 2D-DWT. 虽然 Daubechies 9/7 是以 Daubechies 命名的，但它和普通的 DB 族小波基都不同：普通的 DB 小波基是非对称非正交小波基，而 Daubechies 9/7 是双正交对称小波基，在查阅官方的小波库文档之后，重新按照双正交对称小波基的参数进行排列，最终实现了基于 DB 9/7（有损情况）和 LG 5/3 滤波器（无损情况）的 2D-DWT 和 2D-iDWT，且速度较原来 *for* 循环的算法有了很大的提升. 代码见函数 *dwt* 和 *idwt*.

5 量化

JPEG2000 中的量化是将小波系数的大量可能的离散取值近似为较少个离散值的过程，即小波子带的样本通过量化过程映射到量化索引中. 本文用 $Q(x)$ 表示对 x 量化所得的值，而反量化表示为 $\hat{x} = \overline{Q^{-1}}(Q(x))$. 非细小的量化本身是不可逆的，所以 JPEG2000 的无损压缩中不包括量化. 但在对小波系数进行编码时，不论有损或无损压缩，我们都必须为编码器提供待编码数值的一致解释，称为界定（ranging）. 有损压缩中，量化本身就是一种界定过程. 而无损压缩需要一段独立程序来完成界定. 代码中的量化的核心函数名为 *quantization_math*，入口函数名逐次为 *quantization* 和 *quantization_helper*.

不同的量化策略对画质和码率的影响很大. 限于个人水平和时间，加上小波系数的“概率分布”难以描述，且为了保险起见（复杂的量化策略会带来复杂的数据结构，而量化过程承上启下，数据结构不容出错）（事实上确实出错了一次，用递归方式修正回来了），我们只在程序中对有损压缩使用了最简单的标量均匀量化器，而无损压缩不经过量化或界定. 均匀量化的性能并非最佳，但我们可以通过优化均匀量化方式来提升相同比特率下的显示效果.

5.1 标量量化

标量量化（SQ）是最简单的处理方式. 它将实轴的一个子集中的每一个元素映射为该子集中的一个特定值. 考虑将实轴划分为 M 个不相交的区间

$$I_q = [t_q, t_{q+1}), q = 0, 1, ..., M-1 \tag{10}$$

其中

$$-\infty = t_0 < t_1 < ... < t_M = +\infty \tag{11}$$

在每个区间内，选择点 x_q 作为 I_q 的输出值，则可将标量量化器理解为从 $\mathbb{R}$ 到 $0, 1, , ..., M-1$ 的映射. 对一个给定的 x，$Q(x)$ 是包含 x 的区间 I_q 的索引. 均匀量化的反量化器由

$$\overline{Q^{-1}}(q) = x_q \tag{12}$$

给出.

5.2 标量量化器的优化：Lloyd-Max 标量量化

若要求 $Q(x)$ 所有可能取值的个数 M 是固定不变的，则我们可以根据某些条件来优化量化过程. 满足这些条件的量化器称为 Lloyd-Max 量化器. 我们假定一个平稳过程具有边缘概率密度函数 f_X，并采用均方误差（MSE）来度量失真. 则有

$$\begin{aligned} \text{MSE} &= E\left[(X-\hat{X})^2\right] \\ &= \sum_{q=0}^{M-1} E\left[(X-\hat{X})^2 | X \in I_q\right] P(X \in I_q) \\ &= \sum_{q=0}^{M-1} \int_{t_q}^{t_{q+1}} (x-\hat{x}_q)^2 f_X(x) dx \end{aligned} \tag{13}$$

若要求 $\frac{\partial(\text{MSE})}{\partial t_q} = 0$，即 $(t_q - \hat{x}_{q-1})^2 f_x(t_q) - (t_q - \hat{x}_q)^2 f_x(t_q) = 0$，可解得

$$t_q = \frac{\hat{x}_{q-1} + \hat{x}_q}{2}, q = 1, 2, ..., M-1 \tag{14}$$

类似地，由 $\frac{\partial(\text{MSE})}{\partial \hat{x}_q} = 0$ 解得

$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x f_X(x) dx}{\int_{t_q}^{t_{q+1}} f_X(x) dx}, q = 1, 2, ..., M-1 \tag{15}$$

式 (14) 和 (15) 构成优化标量量化器的必要条件. (14) 表明量化器决策区域的端点应该在输出点的中间（不是“输出点在端点之间”!）. (15) 的分母是 X 位于 I_q 中的概率，由此可得

$$\hat{x}_q = E\left[X|X \in I_q\right] \tag{16}$$

满足这一性质的码字称为“条件均值”或“质心”.

此外，由式 (14) 和 (15) 可得以下性质

$$E[X - \hat{X}] = 0 \tag{17}$$

$$\sigma_{\hat{X}}^2 = \sigma_X^2 - E[(X - \hat{X})^2] \tag{18}$$

$$E[(X - \hat{X})\hat{X}] = 0 \tag{19}$$

$$E[(X - \hat{X})^2|X \in I_j]P(X \in I_j) = E[(X - \hat{X})^2|X \in I_q]P(X \in I_q), \forall j, q \tag{20}$$

这些性质分别说明：量化误差均值为 0（不论是否有 $E(X) = 0$）；量化使数据方差减小的量等于均方误差；量化误差与量化器输出无关（但不是与输入无关）；所有区间对量化误差的贡献相等.

式 (14) 和 (15) 只是必要条件，因此不能保证最优化. 但对于所有对数凹概率密度函数（即 $\log f_X$ 是凹函数或凸函数），最优化是可以保证的 [7]. 幸运的是，均匀分布、拉普拉斯分布和正态分布都满足这一性质. 另外，在均匀量化的基础上，我们可以对某些量化区间再进行均匀划分. 这样就构造了一族嵌入均匀标量量化器. 可以证明，这些量化器全部满足均匀分布的 Lloyd-Max 条件.

5.3 恒域量化

显然，步长为 Δ 的均匀量化

$$I_q = \left[q\Delta - \frac{\Delta}{2}, q\Delta + \frac{\Delta}{2}\right) \tag{21}$$

一般不是最优量化，然而具有均匀区间，且质心码字为 $\hat{x}_q = E[X|X \in I_q]$ 的量化器是非常接近最优量化的 [8]. 对于零均值概率密度函数，如果在均匀量化基础上加宽以 0 为中心的区间，常常可以使 MSE 性能获得改进，并减小量化输出值的熵 $H(\hat{X})$. 被加宽的零均值区间 I_0 有时被称为“储零箱（zero-bin）”. 这种量化称为“恒域（deadzone）均匀标量量化”. 其数学描述为

$$I_q = \begin{cases} [-(1-\xi)\Delta, (1-\xi)\Delta), & q = 0 \\ [(q-\xi)\Delta, (q+1-\xi)\Delta), & q > 0 \\ [(q-1+\xi)\Delta, (q+\xi)\Delta), & q < 0 \end{cases} \tag{22}$$

其中 $\xi < 1$ 决定 I_0 的宽度. 这个量化器的具体实现为

$$q = Q(x) = \begin{cases} \operatorname{sgn}(x)\left\lfloor \dfrac{|x|}{\Delta} + \xi \right\rfloor, & \dfrac{|x|}{\Delta} + \xi > 0 \\ 0, & \text{其他} \end{cases} \tag{23}$$

JPEG2000 标准要求小波子带样本 b 可被恒域量化（b 可代表 LL,LH,HL,HH），且中央量化间隔（即恒域）是其他量化间隔的 2 倍. 此时有 $\xi = 0$，即

$$q_b[n] = \operatorname{sgn}(y_b[n]) \left\lfloor \frac{|y_b[n]|}{\Delta_b} \right\rfloor \tag{24}$$

其中 $y_b[n] \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ 表示子带 b 的样本，而 $q_b[n]$ 表示其量化索引. 每个子带量化时的步长为

$$\Delta_b = 2^{-\epsilon_b} \left(1 + \frac{\mu_b}{2^{11}}\right) \tag{25}$$

其中 ϵ_b 和 μ_b 都是非负整数. 范围为

$$0 \le \epsilon_b < 2^5, 0 \le \mu_b < 2^{11} \tag{26}$$

5.4 界定

为便于表示，首先令

$$\chi_b[n] = \operatorname{sgn}(y_b[n]) \tag{27}$$

$$v_b[n] = \left\lfloor \frac{|y_b[n]|}{\Delta_b} \right\rfloor \tag{28}$$

界定的目的是在编码器与解码器中取相同的位数，该位数足够显示每个子带 b 的量化索引大小 $v_q[n]$. 将足够位数记为 K_b^{max}. 此外，对于超出标称范围的小波子带样本值，引入整数 G，使所有子带样本满足

$$-2^{G-1} < y_b[n] < 2^{G-1}, \forall b \tag{29}$$

参数 G 称为保护位（guard bits）的数目，取值范围 0 到 7. G 取 0 时，有 $y_b[n] \in \left(-\frac{1}{2}, \frac{1}{2}\right)$. 但事实上很少出现 G 可为 0 的情况. 更常见的是 $G = 1$. 而当使用 CDF 9/7 小波核时，$G = 1$ 足以保证式 (29) 的成立. 由 (25) 和 (29) 可得，索引大小必须满足

$$v_b[n] = \left\lfloor \frac{|y_b[n]|}{\Delta_b} \right\rfloor < 2^{\epsilon_b + G - 1} \tag{30}$$

于是

$$K_b^{max} = \max\{0, \epsilon_b + G - 1\} \tag{31}$$

5.5 反量化

假设子带样本 y 已经分配了一个量化索引 $q = \chi v$. 反量化的目的是分配一个重建值 $\hat{y} \in I_q$. 在恒域标量量化情况下，重建过程为

$$\hat{y}_b[n] = \begin{cases} 0, & v_b[n] = 0 \\ \chi_b[n](v_b[n] + \delta_{b,v})\Delta_b, & v_b[n] \neq 0 \end{cases} \tag{32}$$

参数 $\delta_{b,v}$ 可在 $[0, 1)$ 之间取任何值. 理想情况下 $\delta_{b,v}$ 应使 $\hat{y}$ 为 I_q 的统计质心，但是反量化时访问每一个小波子带的基本统计量是不现实的（解压缩的反量化过程中并不知道小波子带是怎样的!）. $\delta_{b,v} = \frac{1}{2}$ 是一个保险的选择，对应中间点重建. 但经验表明，选择较小的值（例如 $\frac{3}{8}$）可获得一些较小的改善，尤其对于高频小波子带而言.

5.6 无损压缩中的界定和重建

无损压缩采用整数小波变换，而量化索引与整数小波子带样本相等，即 $q_b[n] = y_b[n]$. 这些整数样本的标称范围可从原始图片 Y'DbDr 样本位数和小波核的标称增益来确定. 选择合适的保护位个数 G（典型值为 1），使得所有子带样本满足

$$-2^{B-1+X_b+G} < y_b[n] < 2^{B-1+X_b+G} \tag{33}$$

其中 X_b 为小波核的子带 b 的标称增益，即

$$X_{LL} = 0, X_{LH} = X_{HL} = 1, X_{HH} = 2 \tag{34}$$

这样，需要

$$K_b^{max} = B - 1 + X_b + G \tag{35}$$

的位数才足够表示 $v_b[n]$.

无损情况下，界定的重建过程为

$$\hat{y}_b[n] = \begin{cases} 0, & v_b[n] = 0 \\ \chi_b[n](v_b[n] + \lfloor 2^{p_b[n]}\delta \rfloor)\Delta_b, & v_b[n] \neq 0 \end{cases} \tag{36}$$

其中 $p_b[n]$ 表示索引大小的最低位数.

6 内嵌区段编码

6.1 基本编码算法

零编码（zero coding）

该编码根据待编码的数据比特周围的 8 个相邻数据重要性情况生成上下文（CX）. 如表 6.1所示，D0、D1、D2、D3 分别表示待编码数据比特 X 对角线的数据状态值；V0、V1 分别表示 X 垂直方向的状态值，H0、H1 分别表示 X 平方向的状态值. 通过计算 $\sum H_i, \sum V_i, \sum D_i$ 的数值，对 X 进行基于子带的编码. 返回 D 和 Context，D 为 X 的值，Context 的编码如图 6.6所示. 代码见函数 *ZeroCoding*.

D_0	V_0	D_1
H_0	X	H_1
D_2	V_1	D_3

表 6.1 待编码数据的 8 个相邻数据

符号编码 (Sign Coding)

该编码根据待编码的数据比特上、下、左、右的 4 个相邻数据重要性情况以及待编码数据的符号位编码，返回符号编码（XORbit）和 Context. 具体编码规则如表 6.2所示. 其中“1”表示在垂直或者水平方向的相邻两个数据都为重要并且符号都为正；或者只有一个是重要的情况. “0”表示两个方向的相邻数据中两个数据都不重要或者都为重要但是具有不同的符号. -1 表示的情况和 1 的情况相反. 代码见函数 *SignCoding*.

LL and LH subbands (vertical high-pass)			HL subband (horizontal high-pass)			HH subband (diagonally high-pass)		Context label[a]
ΣH_i	ΣV_i	ΣD_i	ΣH_i	ΣV_i	ΣD_i	$\Sigma(H_i+V_i)$	ΣD_i	
2	x[b]	x	x	2	x	x	≥3	8
1	≥1	x	≥1	1	x	≥1	2	7
1	0	≥1	0	1	≥1	0	2	6
1	0	0	0	1	0	≥2	1	5
0	2	x	2	0	x	1	1	4
0	1	x	1	0	x	0	1	3
0	0	≥2	0	0	≥2	≥2	0	2
0	0	1	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

图 6.6　零编码 (zero coding) 上下文 (context) 编码方式

幅度精炼编码 (Magnitude Refinement Coding)

根据该数据位是否被第一次精炼，以及 $\sum H_i + \sum V_i + \sum D_i$ 的值进行编码，返回 D 和 Context．D 为 X 的值，Context 的编码规则如表 6.3所示．代码见函数 *MagnitudeRefinement-Coding*.

游程编码 (Run Length Coding)

仅当一个编码列 (4 个比特数据) 的目前状态都为不重要，且相邻元素也都为不重要时，开始进行游程编码处理．这时，如果一列中的 4 个数据也为不重要数据，则统一编码为一个上下文编码 CX=17 和编码数据 D=0．如果 4 个数据中至少有一个变为重要，则首先将其表示一个上下文 CX=17 和编码数据 D=1．然后，编码 4 个数据中的第一个重要数据的位置信息 (00 11) 作为 D，并编码上下文为 CX=18．此后，编码第一个重要数据的符号位，之后数据的编码按照零编码算法进行．具体编码规则如表 6.4所示．代码见函数 *RunLengthCoding*.

6.2　分流通道

重要性传播编码通道

在此通道中，编码位平面中的目前状态为“无效态”，但有很大概率成为“有效态”的比特数据．只要四周的 8 个比特数据有一个是重要的，就将被进行零编码 (Zero Coding)．在此基础上，如果该位的比特数为 1，再进行符号编码．流程图如图 6.7所示．代码见函数 *SignifiancePropagationPass*.

Horizontal contribution	Vertical contribution	Context label	XORbit
1	1	13	0
1	0	12	0
1	-1	11	0
0	1	10	0
0	0	9	0
0	-1	10	1
-1	1	11	1
-1	0	12	1
-1	-1	13	1

表 6.2 符号编码 (XORbit) 和上下文 (context) 编码方式

是否第一次幅度精炼编码	$\sum H_i + \sum V_i + \sum D_i$	Context
否	x	16
是	≥ 1	15
是	0	14

表 6.3 幅度精炼编码 (Magnitude Refinement Coding) 上下文 (context) 编码方式

幅度精炼编码通道

在此通道中，编码那些在先前位平面已经被判定为有效态的数据，用“幅度精炼编码”编码. 流程图如图 6.7所示. 代码见函数 *MagnitudeRefinementPass*.

清除编码通道

在此通道中，编码位平面中所有还未编码的数据. 当一列 4 个比特都在此通道中被编码，且这 4 个比特都没有重要的相邻数据，则对 4 个比特采用游程编码 (RLC)，否则分别对每个比特采用零编码 (Zero Coding)，如数据为 1，再进行符号编码 (Sign Coding). 流程图如图 6.8所示. 代码见函数 *CLeanUpPass*.

6.3 编码流程

区段分块

将上一步量化后的频块进一步切成 32×32 的区段，作为内嵌区段编码的对象，各区段之间独立运算. 如频块不能被 32 整除，则用 0 补足. // 问题：用 *0* 补足时是否会增加编码的冗余？

4 个比特数据	D	Context
(0,0,0,0)	[0]	[17]
(1,x,x,x)	[1,0,0] + 符号编码 + 零编码	[17,18,18]+ 符号编码 + 零编码
(0,1,x,x)	[1,0,1] + 符号编码 + 零编码	[17,18,18]+ 符号编码 + 零编码
(0,0,1,x)	[1,1,0] + 符号编码 + 零编码	[17,18,18]+ 符号编码 + 零编码
(0,0,0,1)	[1,1,1] + 符号编码	[17,18,18]+ 符号编码

表 6.4　游程编码 (Run Length Coding)D 和上下文 (context) 编码方式

位元层及符号矩阵

对把编码区段内的数据取绝对值，依照位元深度，从高位元（MSB）到低位元（LSB）分成数个位元层.（量化后的数据范围是 *XX–XX*，故共有 *X* 位）此外，由数据的正负组成符号矩阵，其中 0 表示正数，1 表示负数.

初始化上下文、重要性等矩阵

在对整个区段编码的过程中，需要用到的矩阵及其含义如表 6.5所示.

矩阵（向量）名称	大小	含义
重要性矩阵 (S1)	32*32	表明当前状态是否为重要（用于零编码及游程编码）
精炼标记矩阵 (S2)	32*32	表明该元素是否被精炼（用于精炼编码）
编码标记矩阵 (S3)	32*32	表明该元素是否被编码（用于精炼通道及清除编码通道）
符号矩阵 (signs)	32*32	表明该元素的符号（用于符号编码）
位元层 (bitplanes)	X*32*32	小波变化量化后的 2 进制位元表示
D	N*1	编码输出矩阵
上下文 (CX)	N*1	编码输出上下文矩阵

表 6.5　编码中使用的矩阵说明

逐位元层编码

由高位至低位，对各位元层编码. 对于各位元层，依次按照重要性传播编码通道，幅度精炼编码通道，以及清除编码通道的顺序编码. 在编码各通道时，按照如图 6.9所示的方式扫描. 扫描过程从位平面左上角的数据开始，以 4 个数据为一组，连续扫描第一列的第一组 4 个数据后，然后转向扫描第二列的第一组 4 个数据，如此一直扫描到最后一列的第一组 4 个数据. 然后，转向扫描第一列的第二组 4 个数据，一直到最后一列的第二组 4 个数据. 按照这样的顺序依次扫描整个位平面. 扫描过程中矩阵的更新情况如表 6.6所示. 代码见函数 *codeBlockfun*.

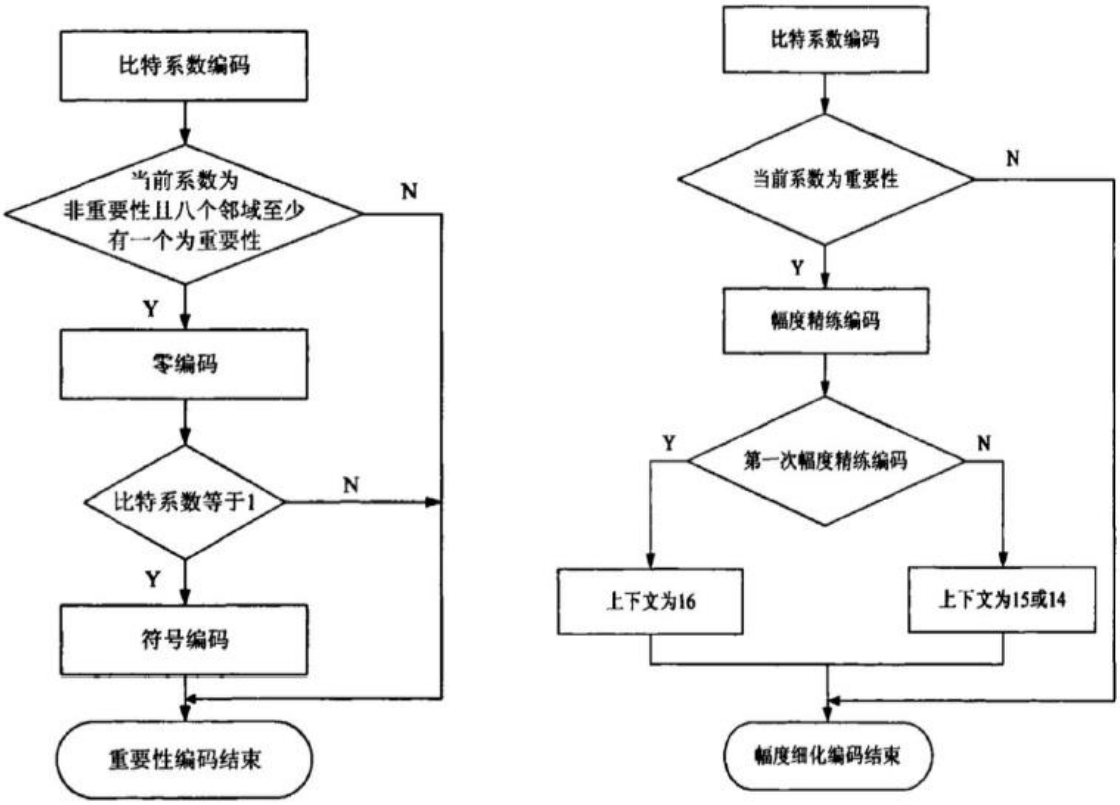

图 6.7 重要性传播编码通道（左）及幅度精炼编码通道（右）编码流程图

7 结果

压缩前后大小比较、图片比较

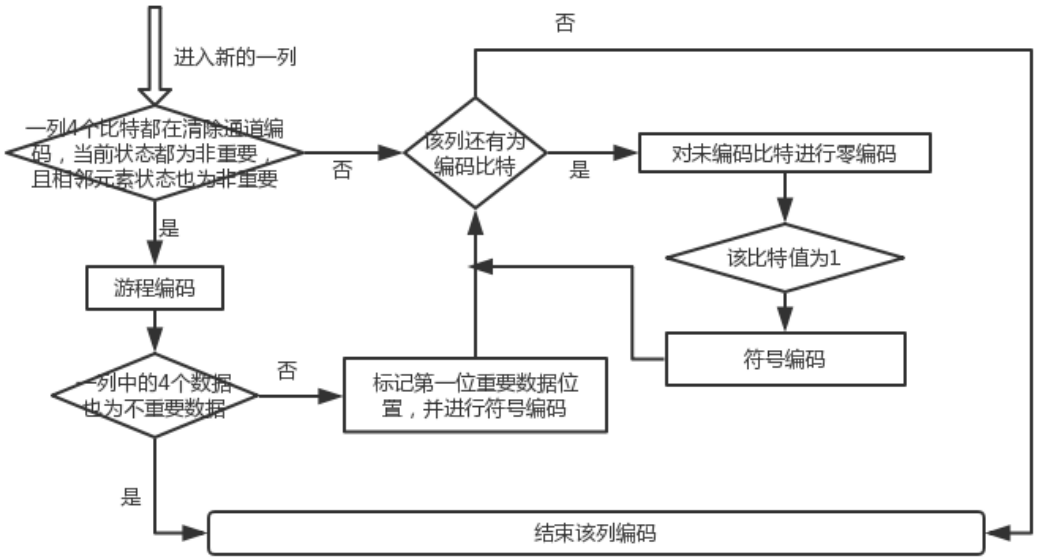

图 6.8　清除编码通道编码流程图

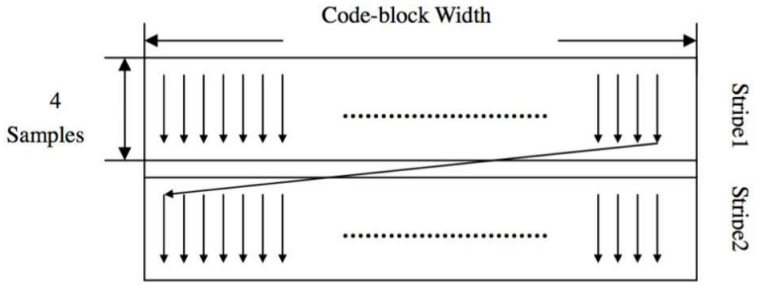

图 6.9　位平面扫描顺序示意图

矩阵（向量）名称	更新时间
重要性矩阵 (S1)	重要性传播编码通道及清除编码通道中更新
精炼标记矩阵 (S2)	精炼通道中更新
编码标记矩阵 (S3)	重要性传播编码通道及精炼通道中更新，结束一层位平面编码清零
D	各通道都更新
上下文 (CX)	各通道都更新

表 6.6　扫描过程中矩阵的更新情况说明

参考文献

[1] jpeg2000. Jordan Van Duyne. https://github.com/jovanduy/jpeg2000.

[2] Taubman D , Marcellin M . JPEG2000 图像压缩基础、标准和实践 [M]. 电子工业出版社, 2004.

[3] CSDN: 压缩算法——JPEG2000 编解码原理. 佚名. https://blog.csdn.net/ytang_/article/details/76571635.

[4] 百度文库:Daubechies 97 和 Le Gall 53 小波分解与合成. 佚名. https://wenku.baidu.com/view/84fc5c7caaea998fcc220e7b.

[5] 知乎: 能不能通俗的讲解下傅立叶分析和小波分析之间的关系? . 佚名. https://www.zhihu.com/question/22864189/answer/40772083.

[6] pywavelets 官方文档. 佚名. https://pywavelets.readthedocs.io/en/latest/index.html.

[7] P. E. Fleischer. Sufficient conditions for achieving minimum distortion in a quantizer. IEEE Int. Conv. Rec., 1:104-111, 1964.

[8] Farvardin N , Modestino J W . Optimum quantizer performance for a class of non-Gaussian memoryless sources[J]. IEEE Transactions on Information Theory, 1984, 30(3):485-497.