

Information Theory Project
JPEG2000 图像压缩实现

王鑫
王鹏
叶涵诚

2019 年 1 月 16 日

第 1 章 JPEG2000

1 概述

JPEG 2000 是基于小波变换的图像压缩标准。JPEG 2000 通常被认为是未来取代 JPEG (基于离散余弦变换) 的下一代图像压缩标准。它改进了 JPEG 在低比特率下的压缩性能, 有更高的压缩率, 对连续色调灰度级和颜色图像可提供技术领域中最先进的压缩性能, 支持按空间位置和图像分量渐进传输等等, 展现了出众的压缩能力。

1.1 实验目的

本项目旨在实现 JPEG2000 标准的核心算法, 并完成图像压缩与解压缩的全过程, 从而对课程信源编码的内容有更好的理解。

1.2 报告结构安排

本报告将按以下三个部分展开。

第一部分, 介绍 JPEG2000 的编码压缩原理, 并较详细地介绍了其实现的方法。

第二部分, 介绍本小组的 JPEG2000 算法实现过程, 以及结果展示。

第三部分, 对实验结果的分析, 以及对该项目的总结与思考, 并提出了一些想法后续的改进之处。

1.3 实现及报告撰写分工

王鹏: 数据预处理及图像分量变换、图像分块与拼接、数据偏移和归一化处理、小波(逆)变换

王鑫: (反)量化、EBCOT(解)编码、Huffman(解)编码

叶涵诚: (反)量化、EBCOT(解)编码、Huffman(解)编码

2 JPEG2000 及其编码算法

2.1 JPEG2000 的编码系统

JPEG2000 编码算法源于 David Taubman 提出的 EBCOT 算法, 使用小波变换, 采用了两层编码策略, 对压缩位流进行分层组织, 不仅提高了压缩效率, 而且压缩码流具有较大的灵活性。JPEG2000 的编解码系统如图 2.1 所示, JPEG2000 的编码系统由七个主要模块组成, 在 JPEG2000 编码过程中, 首先是对原始图像进行离散小波变换, 根据用户要求对变换后小波系数进行量化; 量化后的小波系数划分为小的数据单元——码块, 然后对每个码块进行独立的嵌入式编码; 并将得到的所有码块的嵌入式位流, 按照率失真最优原则分层组织, 形成不同质量的层。对每一层, 按照一定的码流格式打包, 输出压缩码流。下面介绍各部分的作用及基本原理。

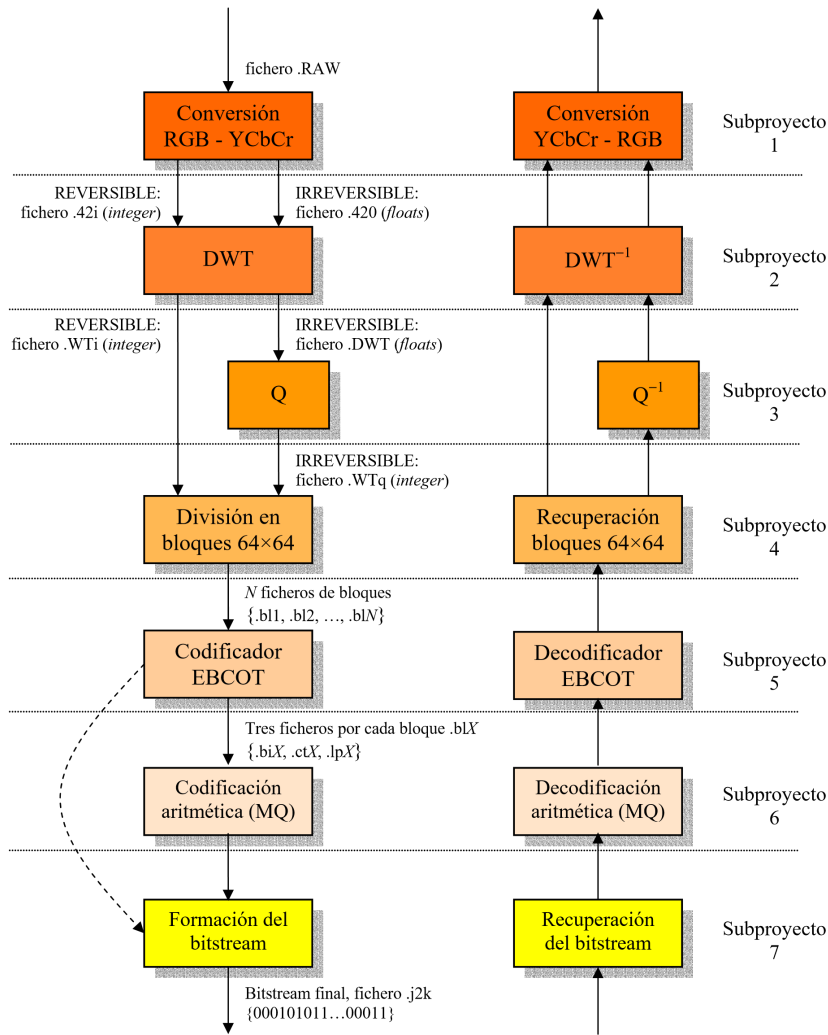


图 2.1 JPEG2000 编解码系统流程.

数据预处理及图像分量变换

JPEG 2000 的核心编码系统的预处理如图2.2所示，主要包括可选的图像分块、数据中心偏移（直流电平平移）、数据的归一化、颜色分量变换.

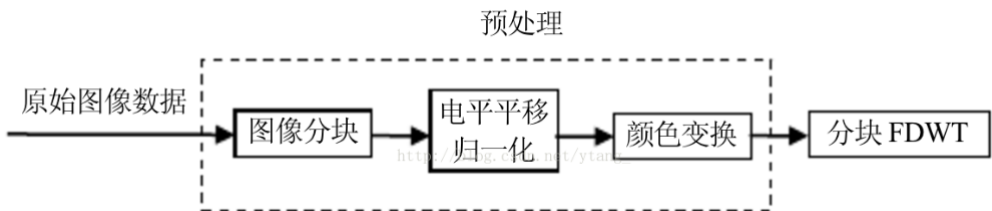


图 2.2 JPEG 2000 的核心编码系统的预处理过程.

图像分块与拼接

与 JPEG 不同，JPEG 2000 算法并不需要将图像强制分成 8×8 的小块. 但为了降低对内存的需求和方便压缩域中可能的分块处理，可以将图像分割成若干互不重叠的矩形块 (tile).

分块的大小任意, 可以整个图像是一个块, 也可以一个像素是一个块. 一般分成 $2^6 \times 2^6$ (即 64×64 像素宽) 的等大方块, 不过边缘部分的块可能小一些, 而且不一定是方的. 在我们实验的过程中发现如果进行分块处理将会得到更高的运算速度, 如果不进行分块处理, 需要的内存和运算量均会比分块处理增加不少.

数据偏移和归一化处理

与 JPEG 的 DCT 一样, JPEG2000 中的 DWT 也需要图像的样本数据关于 0 对称分布, 对 B 位无符号整数的分量样本值 $I(x, y) (0 \leq I(x, y) < 2^B)$ 应该先进行中心偏移 (又称为直流电平平移 DC level shifting) :

$$I'(x, y) = I(x, y) - 2^{B-1} \quad (1.1)$$

对于无损压缩, 因为采用的是整数小波变换, 数据偏移后就可以了. 但是对于有损压缩, 因为采用的是实数型离散小波变换, 所以还需要对偏移后的数据进行归一化处理:

$$I''(x, y) = \frac{I'(x, y)}{2^{-B}} \quad (1.2)$$

在解码的后处理中, 需要将归一化或者平移后的数据进行还原处理:

$$I'(x, y) = 2^B I''(x, y) \quad (1.3)$$

$$I(x, y) = I'(x, y) + 2^{B-1} \quad (1.4)$$

分量变换 (可选)

指对具有多个分量的图像先经过某种变换来降低各分量之间的相关性. 将传统的 RGB (红绿蓝) 色域转换至其他色彩空间. Cr 和 Cb 是差值图像, 直方图在零点附近有很高的峰值.

分量变换是可选的, 只有当图像具有三个或者三个以上的分量, 并且参与变换的三个分量具有相同大小和位深. JPEG2000 定义了两种分量变换, 一种是不可逆的分量变换 (ICT), 即实数的运算; 另一中是不可逆的分量变换 (RCT), 即整数的运算. 在无损压缩中, 只能用 RCT; 在有损压缩中, 两者都可用.

$$ICT : \begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33162 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (1.5)$$

$$iICT : \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.402 \\ 1 & -0.34413 & -0.71414 \\ 1 & -1.772 & 0 \end{pmatrix} \begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix} \quad (1.6)$$

$$RCT : \begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0.25 \\ 1 & -1 & 1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (1.7)$$

$$iRCT : \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & -0.25 & -0.25 \\ 1 & -0.25 & 0.75 \\ 1 & 0.75 & -0.25 \end{pmatrix} \begin{pmatrix} Y \\ Cr \\ Cb \end{pmatrix} \quad (1.8)$$

小波变换

图像的二维离散小波分解和重构过程如图2.3所示，分解过程可描述为：首先对图像的每一行进行 1D-DWT，获得原始图像在水平方向上的低频分量 L 和 高频分量 H，然后对变换所得数据的每一列进行 1D-DWT，获得原始图像在水平和垂直方向上的低频分量 LL、水平方向上的低频和垂直方向上的高频 LH、水平方向上的高频和垂直方向上的低频 HL 以及水平和垂直方向上的的高频分量 HH. 重构过程可描述为：首先对变换结果的每一列进行以为离散小波逆变换，再对变换所得数据的每一行进行一维离散小波逆变换，即可获得重构图像. 由上述过程可以看出，图像的小波分解是一个将信号按照低频和有向高频进行分离的过程，分解过程中还可以根据需要对得到的 LL 分量进行进一步的小波分解，直至达到要求.

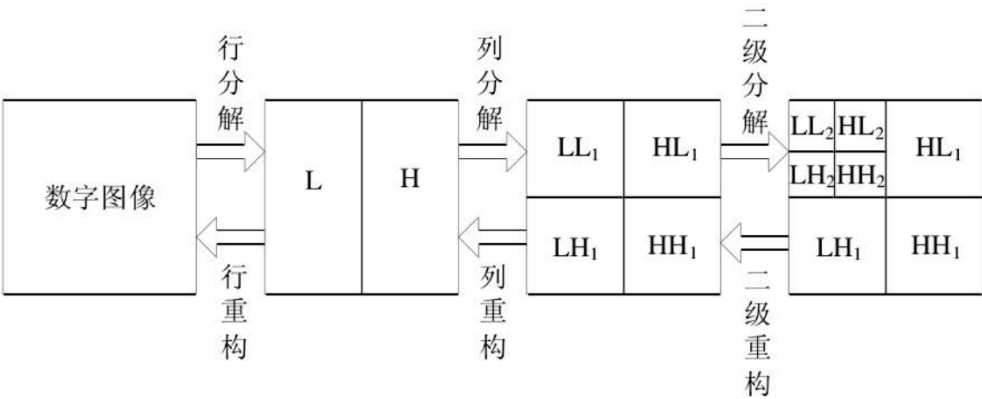


图 2.3 图像的二维离散小波分解和重构过程.

在 JPEG 2000 的核心编码系统中，对有损压缩采用的是基于 Daubechies 9/7 滤波器之提升实现的不可逆 DWT. 对无损压缩采用的则是基于 Le Gall 5/3 滤波器之提升实现的可逆 DWT. JPEG 2000 标准支持基于卷积 (convolution-based) 和基于提升 (lifting-based) 两种滤波模式.

我们在图2.4中给出了在做 db97 提升算法所参考到的公式.

	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 2.4 db97 提升算法.

Daubechies 9/7 滤波器之提升实现如下：

正变换

FDWT 的提升实现包括以下六个步骤：

4 步提升:

$$\begin{cases} y(2n+1) = x(2n+1) - \alpha [x(2n) + x(2n+2)] \\ y(2n) = x(2n) - \beta [y(2n-1) + y(2n+1)] \\ y(2n+1) = y(2n+1) + \gamma [y(2n) + y(2n+2)] \\ x(2n) = x(2n) + \delta [y(2n-1) + y(2n+1)] \end{cases} \quad (1.9)$$

和 2 步缩放:

$$\begin{cases} y(2n+1) = -Ky(2n+1) \\ y(2n) = y(2n)/K \end{cases} \quad (1.10)$$

其中 $\alpha=1.586134342$ 、 $\beta=0.052980118$ 、 $\gamma=0.882911075$ 、 $K=1.230174205$

逆变换

IDWT 的提升实现包括以下六个步骤:

2 步提升:

$$\begin{cases} x(2n+1) = -y(2n+1)/K \\ x(2n) = Ky(2n) \end{cases} \quad (1.11)$$

和 4 步提升:

$$\begin{cases} x(2n) = y(2n) - \delta [x(2n-1) + x(2n+1)] \\ x(2n+1) = x(2n+1) - \gamma [x(2n) + x(2n+2)] \\ x(2n) = x(2n) + \beta [y(2n) + y(2n+2)] \\ x(2n+1) = x(2n+1) + \alpha [x(2n) + x(2n+2)] \end{cases} \quad (1.12)$$

其中 $\alpha=1.586134342$ 、 $\beta=0.052980118$ 、 $\gamma=0.882911075$ 、 $K=1.230174205$

我们在图 2.5 中给出了在做 Le Gall5/3 提升算法所参考到的公式.

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 2.5 Le Gall5/3 提升算法.

Le Gall 5/3 滤波器之提升实现如下:

正变换

$$\begin{cases} y(2n+1) = x(2n+1) - \frac{x(2n)+x(2n+2)}{2} \\ y(2n) = x(2n) + \frac{y(2n-1)+y(2n+1)}{4} \end{cases} \quad (1.13)$$

逆变换

$$\begin{cases} x(2n) = y(2n) - \frac{y(2n-1)+y(2n+1)+2}{4} \\ x(2n+1) = y(2n+1) + \frac{x(2n)+x(2n+2)}{2} \end{cases} \quad (1.14)$$

量化

对变换后的系数进行量化是有损压缩中的重要环节。因为人眼视觉系统对图像分辨率有一定的界限，通过适当的量化减小变换系数的精度，可以在不影响图像主观质量的前提下，进一步提高图像的压缩比。量化的关键是根据变换后的图像的特征、重构的图像的质量要求等因素设定合理的量化步长。经过小波变换，片(tile)分量数据变换成子带形式，若进行 N 级小波分解，可以得到 $3N + 1$ 个子带，每个子带系数都反映了图像不同频率的特征，JPEG2000 标准中采用均匀量化，但不同子带使用不同的量化步长。系数量化的算法为：

$$q_b(u, v) = \text{sign}(a_b(u, v)) \left\lfloor \frac{a_b(u, v)}{\Delta b} \right\rfloor \quad (1.15)$$

其中：

- ① $a_b(u, v)$ 是子带 $b(\text{LL}, \text{LH}, \text{HL}, \text{HH})$ 内的变换系数；
- ② $q_b(u, v)$ 是其量化值， Δb 是量化阶，可由：

$$\Delta b = 2^{R_b - \epsilon_b} \left[1 + \frac{\mu_b}{2^{11}} \right] \quad (1.16)$$

求出。 R_b 取决于信源的编码位数， ϵ_b 和 μ_b 是分配给子带系数的指数和尾数的比特数。在可逆压缩中， $\Delta b = 1$ ， $R_b = \epsilon_b$ ， $\mu_b = 0$ 。

第一、二级编码

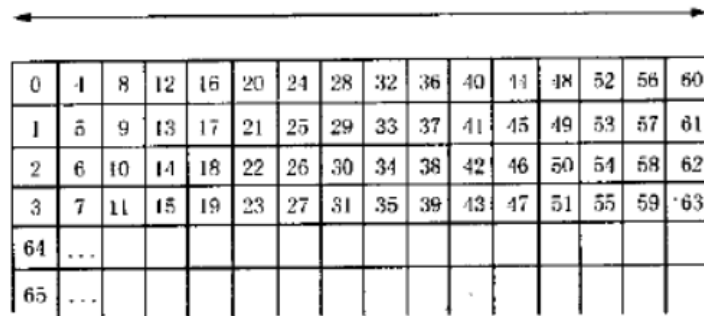
经过量化后的系数要组织成为码块、位平面和编码通道。第一级实质就是对各种编码要素进行了熵编码，主要是基于上下文的 2 元算数编码。经过小波变换和量化，片分量矩阵成为了整数系数的子带矩阵。每个子带又要被划分为大小相同的矩形码块，码块的宽和高都需是 2 的整数次幂，典型值是 32×32 或 64×64 ，本实验采用 64×64 码块，每个码块将被独立编码。每个码块又可以分成位平面，即比特层。从最高有效位平面开始逐平面编码直到最低位平面。

第二级编码过程实际上是分层、打包、形成码流的过程。这层编码完成了 JPEG2000 标准具有的许多优良性能，如压缩质量和分辨率可分级性。分层装配的目的就是按照率失真最优的原则，选取合适的截断点截断每一分块的压缩码流，装配成具有分辨率可伸缩性或质量可伸缩性的满足预定编码长度的最终码流。编码时，对每一个截断点都根据率失真优化算法进行计算。然后将截断点和失真值以压缩的形式同码块位流保存在一起，形成码块的嵌入式压缩位流。

2.2 JPEG2000 系数位平面编码

小波系数量化后的数据以编码块为单位进行熵编码，每个编码块独立编码和同一子带以及其他子带的码块无关。量化后的系数是符号数，每个系数以二进制表示，取每一个系数同一比特位的二进制数构成的二进制数组称为位平面。编码块中的每个位平面以一种特殊的方式进行扫描采样，如图 2.6 所示，从左侧最上面的系数开始，每四个系数为一列，直到扫遍编码块中的所有系数为止。每个位平面中的系数仅以下列三种编码通道之一编码：中药性传播通道、幅度细化通道和清理通道。

3 个编码过程的具体操作由 4 个编码模块执行, 分别是零编码模块, 符号编码模块, 幅度细化编码模块和行程编码模块. 各编码模块产生的是上下文和数据对. 它们在后续的算术编码器中进行编码. JPEG2000 规定了 19 种上下文.



0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63
64	...														
65	...														

图 2.6 编码块中小波系数的扫描模式.

三个通道编码过程

在编码前, 所有的重要性状态位都为 0, 当编码到某一系数的第一个非 0 位时, 该系数对应的重要性状态置为 1.

重要性传播通道

① 条件: 本身的重要性状态为 0, 且 8 个邻居中至少有一个重要性状态为 1. 邻居的定义如图 2.7 所示.

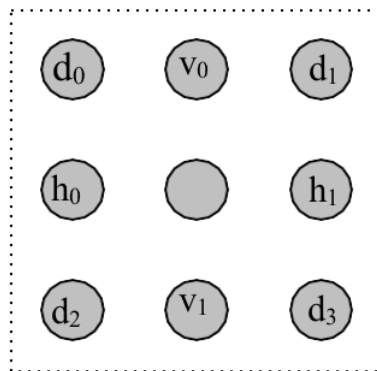


图 2.7 用于形成上下文矢量的 8 邻居系数.

② 操作: 对系数进行零编码模块编码; 如果重要性状态变为 1, 还要对系数符号进行符号编码.

幅度细化通道

对上一位平面已经重要性状态为 1 的系数进行幅度细化编码.

清理通道

对上述两个通道没有进行编码的系数进行编码.

若一系列数都没有被编码且该列系数的所有邻居系数也是不重要的, 则分两种情况:

① 该列有系数在当前位平面变为重要, 则用行程编码编码该列, 并对该系数符号进行符号编码. 该列剩下的系数进行零编码, 如果有系数重要的, 还要对该系数进行符号编码;

② 该列有系数在当前位平面不重要，则用行程编码编码该列。

对于其它情况，则对系数进行零编码，如果系数是重要的，则还要进行符号编码。

四种编码操作

零编码

① 计算编码系数的上下文。根据表2.1、2.2、2.3得到相应的上下文标签。

② 判断系数的重要性。若重要，则输出重要性状态为 1，否则输出重要性状态为 0。

表 2.1 LL 和 LH 子带码块标签索引表

LL 和 LH 子带码块的上下文			标签
Σh	Σv	Σd	
2	x	x	8
1	≥ 1	x	7
1	0	≥ 1	6
1	0	0	5
0	2	x	4
0	1	x	3
0	0	≥ 2	2
0	0	1	1
0	0	0	0

表 2.2 HL 子带码块标签索引表

HL 子带码块的上下文			标签
Σh	Σv	Σd	
x	2	x	8
≥ 1	1	x	7
0	1	≥ 1	6
0	1	0	5
2	0	x	4
1	0	x	3
0	0	≥ 2	2
0	0	1	1
0	0	0	0

符号编码

① 计算待编码系数的符号位上下文标签。上下文取决于系数是否具有显着的垂直或水平邻居，以及这些邻居的符号。使用两个变量（H 和 V），根据邻居（水平或垂直）采用以下值：

表 2.3 HH 子带码块标签索引表

HH 子带码块的上下文		标签
$\Sigma(h+d)$	Σd	
x	≥ 3	8
≥ 1	2	7
0	2	6
≥ 2	1	5
1	1	4
0	1	3
≥ 2	0	2
1	0	1
0	0	0

- 变量中的 +1 表示两个邻居都是重要的并且具有正号.
 - 变量中的 -1 表示两个邻居是显着的和负的.
 - 否则, 在变量中使用值 0.
- ② 根据这些变量的值, 按照表 2.4 中定义的规则应用相应的上下文标签.

表 2.4 符号上下文标签索引表

上下文标签	13	12	11	10	9	10	11	12	13
H	1	1	1	0	0	0	-1	-1	-1
V	1	0	-1	1	0	-1	1	0	-1

幅度细化编码

在细化阶段中对比特位进行编码时, 只有 3 种可能的上下文, 这取决于它是否是第一次改进系数以及它是否具有重要的邻居.

① 若系数第一次在幅度细化通道编码, 且其邻居系数没有一个是重要的, 用上下文标签 14 表示, 否则表示为 15;

② 如果系数已被细化过, 则用上下文标签 16 表示.

行程编码

行程编码仅用一个符号说明一列的 4 个系数, 在前一个位平面都是不重要的系数, 在当前编码位平面的重要状态. 行程编码的条件是: 一列 4 个系数及其所有邻居系数都是不重要, 用上下文标签 17 表示. 若这列系数在当前编码位平面上都是不重要的, 输出符号为 0, 若有一个系数变为重要的则输出符号 1, 这是要中止行程编码, 并从该列第一个变为重要的系数开始对后面的系数进行编码, 并用上下文标签 19 表示该编码系数的位置, 之后输出的符号表示该系数的位置, 同时要对该系数的符号进行符号编码.

2.3 MQ 二值算术编码

二进制算术编码是算术编码中的一种特殊情况. JPEG2000 所用的 MQ 算术编码器属于自适应二进制算术编码器, 它是指编码系统用来划分区间的当前符号概率估计是可以根据

已经传输和编码的信息串调整的。由于本实验采用的是 Huffman 编码方式，故在此不予详细展开 MQ 二值算术编码的原理。

3 实验过程

本实验基于 MATLAB 平台实现 JPEG2000 的压缩算法，根据原理自主编写绝大部分代码。(其中 EBCOT 部分较难,参照了加泰罗尼亚大学作者 *almacbe*(<https://github.com/almacbe/ebcot-code>) 所编写的西班牙语 C++ 文件，Huffman 编码文件参照网上现有代码进行了一定的改编)。

我们实现的是 JPEG2000 的有损压缩方式。

在预处理部分，我们先进行了电平位移，然后进行颜色变换（采用上文所描述的 ICT），其代码如下：

```
1 RGB=(double(x)-128)./256;
2 Y = 0.299*R + 0.587*G + 0.114*B;
3 U = -0.147*R- 0.289*G + 0.436*B;
4 V = 0.615*R - 0.515*G - 0.100*B;
```

在小波变换部分，我们严格按照上文所描述的 Db9/7 提升滤波器进行正变换和反变换并获得了非常好的效果。

```
1 %提升法正变换部分代码示例：
2 for time=1:T; % 行正变换
3 % d;
4 x1(n1,:)=x(n2,:)+d1(2)*x(n2-1,:)+d1(1)*x(1,:);
5 x1([1:n1-1],:)=x([2:2:n2-2],:)+d1(2)*x([1:2:n2-3],:)+d1(1)*x([3:2:n2-1],:);
6 % p;
7 x(1,:)=x(1,:)+p1(2)*x1(n1,:)+p1(1)*x1(1,:);
8 x([2:n1],:)=x([3:2:n2-1],:)+p1(2)*x1([1:n1-1],:)+p1(1)*x1([2:n1],:);
9 x([n1+1:n2],:)=x1([1:n1],:);
10 % d;
11 x(n1+1,:)=x(n1+1,:)+d2(2)*x(n1,:)+d2(1)*x(1,:);
12 x([n1+2:n2],:)=x([n1+2:n2],:)+d2(2)*x([1:n1-1],:)+d2(1)*x([2:n1],:);
13 % p;
14 x(n1,:)=x(n1,:)+p2(2)*x(n1+1,:)+p2(1)*x(n1+2,:);
15 x(n1-1,:)=x(n1-1,:)+p2(2)*x(n2,:)+p2(1)*x(n1+1,:);
16 x([1:n1-2],:)=x([1:n1-2],:)+p2(2)*x([n1+2:n2-1],:)+p2(1)*x([n1+3:n2],:);
17 % 归一
18 x([1:n1],:)=p3*x([1:n1],:); % 原大小
19 x([n1+1:n2],:)=d3*x([n1+1:n2],:); % 一半大小
```

在量化部分，对于量化步长的确定，首先将参数分别设为 7 和 8。对于 Rb 的确定，根据码块中的（绝对）最大值与码块的子带信息分为了 12 种情况。具体代码如下：

```
1 switch subband
2     case 0 %% LL
3         gainb=0;
4     case 1 %% LH
```

```

5         gainb=1;
6     case 2 %% HL
7         gainb=1;
8     case 3 %% HH
9         gainb=2;
10 end
11 if max(max(x))>1
12     RI=2;
13 elseif max(max(x))>1e-2
14     RI=-2;
15 else
16     RI=-4;
17 end
18 Rb=RI+gainb;

```

在 EBCOT 部分, 我们首先将量化好后的数据进行分块, 分成大小为 64×64 的子块, 并通过元胞数组存储每个子块的信息 (如级数, 子带数, 像素集等等), 将元胞数组作为 EBCOT 系统的输入, 每四行分为一个迭代进行三通道编码, 将三个通道编码数据存放在一起输入的 *x.file* 数组中^①, 形成最后的码流。

码元的写入与读取分别通过 *WriteBit.m* 与 *ReadBit.m* 两个文件实现, 其中具体的读写过程较复杂, 参见附录文件中的代码。

在完成 EBCOT 编码后, 生成三个文件 *x.file* (码流), *FLength* (编码信息文件), *Fctxt* (上下文信息文件)。

之后是 Huffman 编码部分, 将码流作为输入, 由于 Huffman 编码的对象为文本类型或字符串类型, 因此须将码流转换为字符串后再输入, 这样以后就得到了最终的压缩码流。

至于为什么没有采用原装的二值算术编码 (MQ 编码) 来实现这一环节, 原因主要是由我们 EBCOT 部分的输出结果并不是标准的二进制码流^②, 因此很难使用二值算术编码器进行编码, 于是选择采用简单的 Huffman 编码方式进行编码, 即待码流全部生成后, 根据各符号的概率或频率进行排序, 再通过 Huffman 数对各符号进行长度不等的二进制编码, 结果显示 Huffman 编码的压缩效率在 0.18 附近。

恢复过程即为相应部分的逆过程的拼接, 需要注意的是在电平位移的恢复环节与小波逆变换两个环节中有一些细节不能忽视。在小波逆变换的过程中, 需要先将分块拼接起来再进行逆变换, 否则会导致恢复出的图像块效应明显。其次, 在电平位移的恢复过程中, 要注意数值的溢出问题, 应先将进行 double 型的运算, 后将其转换为 uint8 型。

最终的恢复图像与原图像的比较如图 3.8 所示。可以看到, 图像恢复效果基本上与原图保持一致。本小组经讨论, 认为压缩率是指传输过程中的图像大小压缩比例, 并非恢复完以后的图像压缩比例。如果是这个意义上的压缩率的话, 本实验压缩效果接近 1/6。

① *x* 为一个结构体, 由 C++ 头文件中的码流文件定义转换而来, 其中包含 *x.input*, *x.output* (这两个为输入字节与输出字节), *x.pos*, *x.quedan* (这两个西班牙文不知道什么意思, 但是在读写码元的时候有用, 因此只能原样照抄) 以及 *x.file* (码字写入与读取的存储位置)

② 具体原因可能是由于参考代码的作者在编写写入比特流函数时使用了十六进制数的位运算, 导致我们没搞明白这样做的原因, 于是依葫芦画瓢将其转换为十进制的位运算, 这使得输出码流为非二值的整数。当然, 也有可能是原代码作者实际就没有输出二进制码流, 我们没有调试过原代码, 所以不清楚是否如此。



图 3.8 恢复图像与原图像的比较结果比较.

4 实验分析与总结

本次实验基本实现了 JPEG2000 有损压缩的整个过程, 并研究了其核心部分小波变换与 EBCOT 编码过程. 其中, 有部分环节采用的是与原标准不同的方式进行处理 (如, 最终的码流形成环节, 量化参数的确定).

对于实验中的量化环节, 其参数值对后续的编码过程会产生重要影响, 因为从信道角度来考虑的话, 这相当于不同的加性噪声功率对三个扫描通道的影响, 导致编码效率的不同, 尤其是死区 (deadzone) 的长度对 HH 区域的编码效率影响会很大. 在 HH 区域, 小波系数均趋近于 0, 若 deadzone 的宽度很宽, 则在对 HH 区域的编码过程中, 利用行程编码可以很高效地进行编码. 但是太宽会造成低频部分, 即 LL 部分地大量信息损失, 因此采用多步长的量化策略.

在 EBCOT 编码环节中, 结构的利用虽然增加了可读性, 但是影响了编码的效率与速度. 事实上, 在我们所转换过来的代码文件中, 存在部分冗余, 我们已经尽可能地改善了从 C++ 文件转换为 MATLAB 文件时存在的代码冗余, 但仍不可避免地存在冗余. 比如读写码流的文件, 由于不懂西班牙文, 导致结构 x 不知道如何简化, 使得该部分的阅读存在一定障碍, 但好在只是影响编码的运行时间, 并没有造成代码的错误出现.

对于压缩结果的分析, 可能是由于量化步长的设定不够灵活造成压缩效果的不明显. 另外, 我们对 EBCOT 编码的部分在输出的码流形式上还有一些疑惑, 这可能也是一个原因. 最后, 分块的大小对最后的压缩结果也会有一定的影响, 这主要体现在分块数据范围的不同对 EBCOT 编码的分辨率会产生影响. 同时, 分块的大小也会影响编码的运行时间, 因此

合适的分块才能起到好的压缩效果.

对于实验的后续改进,可以考虑从无损与有损压缩的比较,还有使用 MQ 编码器与 Huffman 编码器的压缩效率的比较去进一步的研究与探讨.

参考文献

- [1] JPEG2000 图像压缩基础、标准和实践. David S. Taubman 等著. 魏江力等译. 电子工业出版社 (2004)
- [2] JPEG2000 图像压缩原理及实现. 黄紧德. <https://www.docin.com/p-1633534858.html>.
- [3] 数据压缩导论. Khalid Sayood 著. 贾洪峰译. 人民邮电出版社 (2014)
- [4] *almacbe* 的 github 中关于 ebcot 的代码.<https://github.com/almacbe/ebcot-code>
- [5] Daubechies97 和 LeGall53 小波分解与合成. <https://wenku.baidu.com/view/84fc5c7caaea998fcc220e7b.html>