JPEG 图像解码方案

吴嘉慧

(中山大学信息科学与技术学院计算机系，广州 510006)

摘 要：JPEG 文件是当前网络最为流行的图像文件格式，具有高压缩比、多种质量选择等特征。从解码角度详细分析 JPEG 压缩算法，而非常规的从编码的角度，给出实践过程中遇到的问题和解决方法。

关键词：JPEG 解码；图像压缩算法；行程编码；离散余弦变换；哈夫曼编码

引 言

JPEG(Joint Photographic Experts Group)是联合图像专家小组的英文缩写，负责制定静态数字图像的编码标准。该小组开发出连续色调、多级灰度、静止图像的数字图像压缩编码方法，即 JPEG 算法。用 JPEG 算法压缩出来的静态图片文件称为 JPEG 文件，扩展名通常为 *.jpg、*.jpe*.jpeg。

对于图像压缩，JPEG 专家组开发了多种压缩算法、编码方法和编码模式。在实际应用中，JPEG 图像使用的是离散余弦变换、哈夫曼编码和顺序模式。

由于具有高压缩比、多种质量选择等良好特征，JPEG图像格式已经在网络上流行多年。JPEG 编解码中文资料大都从编码角度分析。如果从编码角度学习，很可能由于利用已有图像解码软件无法显示编码生成的图片而陷入困境，而从解码角度认识 JPEG 格式，由于已存在一个正确的编码图像，通过用解码的图像与正确图像对比，从而更容易发现自身错误，修正对 JPEG 格式的理解。

本文从解码角度详细分析 JPEG 压缩算法，并且给出实践过程中遇到的问题和解决方法，展示一个宏观的 JPEG 数据组织方式，列举一些解码中的细节事项。

1 解码过程的概述

JPEG 图像文件解码步骤大概如下：

●从文件头读出文件的相关信息。JPEG 文件数据分为文件头和图像数据两大部分，其中文件头记录了图像的版本、长宽、采样因子、量化表、哈夫曼表等重要信息。所以解码前必须将文件头信息读出，以备图像数据解码过程之用。

●从图像数据流读取一个最小编码单元(MCU)，并提取出里边的各个颜色分量单元。关于如何从数据流中把一个个连续存储的 MCU 分割开来，以及如何从各个 MCU 中将多个颜色分量分割开来。

●将颜色分量单元从数据流恢复成矩阵数据。利用文件头给出的哈夫曼表，对分割出来的颜色分量单元进行解码，把其恢复成 8×8 的数据矩阵。

●8×8 的数据矩阵进一步解码。此部分解码工作以 8×8 的数据矩阵为单位，其中包括相邻矩阵的直流系数差分解码、利用文件头给出的量化表反量化数据、反 Zig-zag 编码、隔行正负纠正、反向离散余弦变换等 5 个步骤，最终输出仍然是一个 8×8 的数据矩阵。

●颜色系统 YCrCb 向 RGB 转换。将一个 MCU 的各个颜色分量单元解码结果整合起来，将图像颜色系统从 YCrCb 向 RGB 转换。

●排列整合各个 MCU 的解码数据。不断读取数据流中的 MCU 并对其解码，直至读完所有 MCU 为止，将各 MCU 解码后的数据正确排列成完整的图像。

2 读入文件的相关信息

按照 JPEG 文件数据存储方式，把要解码的文件的相关信息读出。参考方法是设计一系列的结构体对应各个标记，并存储标记内的信息。以下给出读取过程中最容易出错的哈夫曼树的建立过程。

(1)读出哈夫曼表数据

①理论说明

在标记段 DHT 内，包含了一个或者多个的哈夫

曼表。对于单个哈夫曼表,应该包括三部分:

●哈夫曼表 ID 和表类型:

0x00 表示 DC 直流 0 号表;

0x01 表示 DC 直流 1 号表;

0x10 表示 AC 交流 0 号表;

0x11 表示 AC 交流 1 号表。

●不同位数的码字数量

JPEG 文件的哈夫曼编码只能是 1~16 位。这个字段的 16 个字节分别表示 1~16 位的码字的个数。

●编码内容

这个字段记录了哈夫曼树中各个叶子结点的权。所以,上一字段的 16 个数值之和就应该是本字段的长度,也就是哈夫曼树中叶子结点个数。

②举例说明

以下面一段哈夫曼表数据举例说明(数据全部以 16 进制表示):

11 00 02 02 00 05 01 06 01 00 00 00 00 00 00 00 00

00 01 11 02 21 03 31 41 12 51 61 71 81 91 22 13 32

第一个划线部分为哈夫曼表 ID 和表类型,其值 0x11 表示此部分描述的是 AC 交流 1 号表;第二个划线部分为不同位数的码字的数量,具体意义为:没有 1位和 4 位的哈夫曼码字,2 位和 3 位的码字各有 2 个,5 位码字有 5 个,6 位和 8 位码字各有 1 个,7 位码字各有 6 个,没有 9 位或以上的码字;第三个划线部分为编码内容。由第二划线部分数据知道,此哈夫曼树有 17 个叶子结点,即本字段应该有 17 个字节。这段数据表示 17 个叶子结点按从小到大排列,其权值依次为 0、1、11、2、21、3、31、41……

(2)建立哈夫曼树

①理论说明

建立哈夫曼树的具体方法为:

●第一个码字必定为 0。

如果第一个码字位数为 1,则码字为 0;

如果第一个码字位数为 2,则码字为 00;

如此类推。

●从第二个码字开始,

如果它和前面的码字位数相同,则当前码字为它前面的码字加 1;

如果它的位数比它前面的码字位数大,则当前码字是前面的码字加 1 后再在后边添若干个 0,直至满足位数长度为止。

②举例说明

继续以上面的例子说明问题。

●由于没有 1 位的码字,所以第一个码字的位数为 2,即码字为 00;

●由于 2 位的码字有两个,所以第二个码字位数仍为 2,即码字为 00+1=01;

●第三个码字为 3 位,比第二个码字长 1 位,所以第三个码字为:01+1=10,然后再添 1 个“0”,得 100;

●……

3 初步了解图像数据流的结构

(1)理论说明

分析图像数据流的结构,以一个从宏观到微观的顺序来作详细剖析,即:

数据流→最小编码单元→数据单元与颜色分量→颜色分量单元。

●在图片像素数据流中,信息可以被分为一段段最小编码单元(Minimum Coded Unit,MCU)数据流。所谓 MCU,是图像中一个正方矩阵像素的数据。这些矩阵的大小是这样确定的:

查阅标记 SOF0 得到图像不同颜色分量的采样因子。大多图片的采样因子为 4:1:1 或 1:1:1。记三个分量中水平采样因子最大值为 Hmax,垂直采样因子最大值为 Vmax,则单个 MCU 矩阵的宽就是 Hmax×8 像素,高就是 Vmax×8 像素。

如果整幅图像的宽度和高度不是 MCU 宽度和高度的整数倍,那么编码时会用某些数值填充进去,保证解码过程中 MCU 的完整性。

另外,在数据流中,MCU 的排列方法是从左到右,从上到下。

●每个 MCU 又分为若干个数据单元。数据单元的大小必定为 8×8。

JPEG 文件与 BMP 文件有所不同,它把图片分成 Y、Cr、Cb 三张子图,然后分别压缩。三个颜色分量的采样因子可能一样(如 1:1:1),也可能不一样(如 4:1:1)。

每个 MCU 内部,数据的顺序是 Y、Cr、Cb。如果一个颜色分量有多个数据单元,则顺序是从左到右,从上到下。

(2)举例说明

下面通过一幅 32px×35px 的图像,对上面两个问题作具体说明。

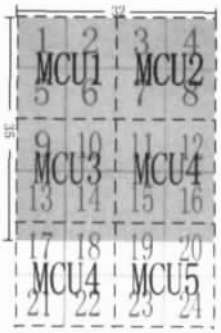

图 1 整张完整的图像

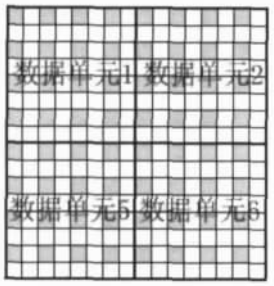

图 2 将图像的 MCU1 放大

图 1 中灰色部分为实际图像大小（32px×35px）；粗虚线表示各个 MCU 的分界；细虚线表示 MCU 内部数据单元的分界。

假设此图的采样因子为 4:1:1，即（2×2):(1×1):(1×1)。此时，Hmax=2，Vmax=2。所以，MCU 的宽为 16 像素，高为 16 像素。图像实际的宽刚好是 2 个 MCU，但高则稍稍大于 2 个 MCU 的高度，所以要补足 3 行 MCU。

在数据流中，MCU 的顺序是 MCU1→MCU2→MCU3→MCU4→MCU5→MCU6。

每个 MCU 又分为 4 个数据单元。采样因子 4:1:1 表示 Y 分量的水平和垂直方向都是每 2 个像素采样 2 次；Cr 分量和 Cb 分量的水平和垂直方向都是每 2 个像素采样 1 次。因此，Y 分量取满 256 个采样点；Cr 分量和 Cb 分量各自只有 64 个采样点，取法如图 2 的灰色点。

如果以 MCU1 说明 MCU 数据的次序，则依次为 Y_1、Y_2、Y_5、Y_6、Cr_{1256}、Cb_{1256}。图 2 中全部 256 个点均是 Y 的采样点，灰色部分为 Cr 分量和 Cr 分量的采样点。

对于整张图片来说，数据流的数据依次是：

[Y_1、Y_2、Y_5、Y_6、Cr_{1256}、Cb_{1256}]、[Y_3、Y_4、Y_7、Y_8、Cr3478、Cb_{3478}]、[Y_9、Y_{10}、Y_{13}、Y_{14}、$Cr_{9101314}$、$Cb_{9101314}$]、……

4 颜色分量单元的内部解码

（1）理论说明

‘颜色分量单元’约定为说明问题而建立的概念，指的是 MCU 中某个颜色分量中的一个 8×8 数据块，例如上面提到的 Y_1、Cr_{1256}、Cb_{1256} 都是一个颜色分量单元。

图像数据流是以位（bit）为单位存储信息的，并且内部的数据都是在编码时通过正向离散余弦变换（FDCT）得到的结果，所以颜色分量单元应该由两部分组成：1 个直流分量和 63 个交流分量。

解码的过程其实就是哈夫曼树的查找过程。

首先查阅标记段 SOS 中的颜色分量信息，可以得出各个颜色分量对应使用的直流分量和交流分量使用的哈夫曼树编号。一般来说，

Y 分量：直流分量：直流 0 号哈夫曼树，交流分量：交流 0 号哈夫曼树；

Cr 分量：直流分量：直流 1 号哈夫曼树，交流分量：交流 1 号哈夫曼树；

Cb 分量：直流分量：直流 1 号哈夫曼树，交流分量：交流 1 号哈夫曼树。

颜色分量单元内部综合运用了 RLE 行程编码和哈夫曼编码来压缩数据。每个像素的数据流由两部分构成：编码和数值。具体读入单个颜色分量单元的步骤如下：

①从此颜色分量单元的起点为单位读入，直到读入编码与该分量的直流哈夫曼树的某个码字一致，然后用查得该码字对应的权值。权值表示该直流分量数值的二进制位数，也就是接下来需要读入的位数。

②继续读入位数据，直到读入的编码与该分量交流哈夫曼树的某个码字一致，然后查得该码字对应的权值。权值的高 4 位表示当前数值前面有多少个连续的零，低 4 位表示该交流分量数值的二进制位数，也就是接下来需要读入的位数。

③不断重复步骤②，直到满足交流分量数据结束的条件。结束条件有两个，只要满足其中一个即可：

●当读入码字的权值为零，表示往后的交流变量全部为零；

●已经读入 63 个交流分量。

④各个数值的译码按表 1 进行。

表 1 数值编码对照表

编码长度	编码数值（二进制）	实际数值（十进制）
1	0,1	-1,1
2	00,01,10,11	-3,-2,2,3
3	000,001,010,011,100,101,110,111	-7,-6,-5,-4,4,5,6,7
4	0000,……,0111,1000,……,1111	-15,……,-8,8,……,15
5	00000,……,01111,10000,……,11111	-31,……,-16,16,……,31
6	……	-63,……,-32,32,……,63
7	……	-127,……,-64,64,……,127
8	……	-255,……,-128,128,……,255
9	……	-511,……,-256,256,……,511
10	……	-1023,……,-512,512,……,1023
11	……	-2047,……,-1024,1024,……,2047
12	……	-4095,……,-2048,2048,……,4095
13	……	-8191,……,-4096,4096,……,8191
14	……	-16383,……,-8192,8192,……,16383
15	……	-32767,……,-16384,16384,……,32767

（2）举例说明

某个颜色分量单元数据如下：

D3 5E 6E 4D 35 F5 8A

若以二进制表示，并假设该颜色分量单元对应以下直流哈夫曼树（表 2）和交流哈夫曼树（表 3），则可将各个以位为单位的数据流拆分如下：

110 1001101 01 1 11001 101 11001 001 101 00 11010 1 1111010 11 00 01010

表 2 直流哈夫曼树

序号	码字长度	码字	权值
1	2	00	0x00
2	2	01	0x01
3	2	10	0x02
4	3	110	0x07
5	4	1110	0x1e

表 3 交流哈夫曼树

序号	码字长度	码字	权值
1	2	00	0x00
2	2	01	0x01
3	3	100	0x11
4	3	101	0x02
5	5	11000	0x21
6	5	11001	0x03
7	5	11010	0x31
8	5	11011	0x41
9	5	11100	0x12
10	6	111010	0x51
11	7	1110110	0x61
12	7	1110111	0x71
13	7	1111000	0x81
14	7	1111001	0x91
15	7	1111010	0x22

读入数据流并对照直流哈夫曼树，第一个哈夫曼编码为 110，其权值为 7，所以往后读入 7 位数据 1001101，译码成数值为 77。因为颜色分量单元只有一个直流分量，所以下一个是首个交流分量。

继续读入数据流并对照交流哈夫曼树，得哈夫曼编码为 01，其权值为 1，所以它的前面没有零，并往后读入 1 位数据 1，译码成数值为 1。如此类推，最后读到哈夫曼编码 00，其权值为 0，满足交流变量结束条件。最后剩余的 01010 对本颜色分量单元来说是冗余的，它可能属于下一个颜色分量单元。

实际上，这段数据译码为：

77，(0，1)，(0，5)，(0，-6)，(0，-3)，(5，1)，(2，3)

因此，把它置于 1 个 8×8 的矩阵中如表 4。

表 4 译码后的 8×8 矩阵

77	1	5	-6	-3	0	0	0
0	0	1	0	0	3	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

5 直流系数的差分编码

每一种颜色分量内，相邻的两个颜色分量单元的直流变量是以差分来编码的。也就是说，通过步骤 3 解码出来的直流变量数值只是当前颜色分量单元的实际直流变量减去前一个颜色分量单元的实际直流变量。而实际的交流变量应该为：

$DC_n=DC_{n-1}+Diff$

其中 Diff 为差分校正变量，也就是直接解码出来的直流系数。

注意，3 个颜色分量的直流变量是分开进行差分编码的。也就是说，1 张图片有 3 个直流校正变量。另外，当数据流中出现标记 RSTn，则 3 个颜色分量的直流差分校正变量 Diff 都需要重新复位到 0。

6 反量化

不同的颜色分量使用不同的量化表，这个可以从标记段 SOF 中的颜色分量信息字段查得。一般是 Y 分量使用量化表 0，而 Cr、Cb 两个分量共同使用量化表 1。

反量化的过程比较简单。只需要对 8×8 的颜色分量单元的 64 个值逐一乘以对应的量化表内位置相同的值即可。图像内全部的颜色分量单元都要进行反量化。

7 反 Zig-zag 编码

如果将反量化后的每个 8* 8 颜色分量单元的每个元素编号，如下图 3，那么反 Zig-zag 编码的过程就是把矩阵元素按图 4 重新排列。

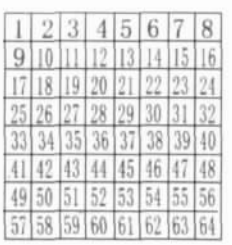

图 3 将颜色分量单元元素编码

图 4 反 Zig-zag 编码

关于量化和反 Zig-zag 编码的先后顺序，不同资料有不同的见解。经过实践，解码的过程中应该直接用文件提供的量化表反量化矩阵数据，再将其反 Zig-zag 编码才能正确解码。

8 隔行的正负纠正

通过对已知图像进行 JPEG 编码压缩，然后和该图的 JPEG 文件数据对比发现的问题。

实际上，就是必须对每个颜色分量单元的奇数行（每个颜色分量单元有 8 行，假设把它按 0、1、……、6、7 编出行号），即 1、3、5、7 行，进行取相反数操作（正的变负，负的变正）。

9 反向离散余弦变换

文件中的数据是在编码时通过正向离散余弦变换得到的结果，所以现在解码就必须将其反向离散余弦变换(IDCT)，就是把颜色分量单元矩阵中的频率域数值向时空域转换。

设正负纠正后的频率域矩阵为 F[u][v]，反向离散余弦变换后的矩阵为 f[i][j]，其中 0≤u,v,i,j≤7。公式如下：

$$f[i][j]=\frac{1}{4}[\sum_{u=0}^{7}\sum_{v=0}^{7}(C(u)C(v)\cdot f[u][v]\cdot\frac{(2i+1)u\pi}{16})]$$

其中：

$$C(u)=\frac{1}{\sqrt{2}}(当\ u=0),C(u)=1(当\ u\neq 0);$$

$$C(v)=\frac{1}{\sqrt{2}}(当\ v=0),C(u)=1(当\ v\neq 0);$$

10 YCrCb 向 RGB 转换

要在屏幕上显示图像，就必须以 RGB 模式表示图像的颜色，所以，解码时需要把 YCrCb 模式向 RGB 模式转换。

正如前面提到，并不是每种颜色分量的采样因子都一样，所以转换时需要注意。由本文第 3 节对 4:1:1 的采样因子的分析，可以知道一个 MCU 里有 4 个 Y 分量单元，而 Cr 分量和 Cb 分量各自只有 1 个分量单元。以图 2 为例，仅有的一个 Cr 分量单元应该平铺用于 4 个 Y 分量单元，即左上角 16 个值用于 Y1，右上角 16 个值用于 Y2，左下角 16 个值用于 Y5，右下角 16 个值用于 Y6。对于 Cb 分量，道理一样。

另外，由于离散余弦变化要求定义域的对称，所以在编码时把 RGB 的数值范围从 [0, 255] 统一减去 128 偏移成[-128, 127]。因此解码时必须为每个分量加上 128。具体公式如下：

R=Y +1.402×Cb+128;

G=Y- 0.34414×Cr- 0.71414×Cb+128;

B=Y +1.772×Cb+128;

还有一个问题，通过变换得出的 R、G、B 值可能超出了其定义域。如果大于 255，则截断为 255；如果小于 0，则截断为 0。

至此，每个 MCU 的解码已经完成。只要将每个 MCU 组成一幅完整的图像就完成了一张 JPEG 图像的解码了。

参考文献

[1]张益贞. Visual C++实现 MPEG/JPEG 编解码技术. 北京：人民邮电出版社, 2002, 11

[2]CCIT.Information Technology- Digital Compression and Conding of Continuous- ton Still Images- requirements and Guidelines.http://www.wotsit.org/download.asp?f=itu-1150PDF (访问日期: 2007- 1- 1)

[3]云风.JPEG 简易文档 V2.11.http://rtornados.bokee.com/2442419.html(访问日期: 2006- 12- 30)

(收稿日期: 2007- 01- 08)

Solution of JPEG Image Decoding

WU Jia- hui

(College of Information Science and Technology,Sun Yat- sen University, Guangzhou 510006 China)

Abstract: JPEG images are widely used in Internet, owing to its high compression ratio and multi- quality. Analyses the JPEG compression algorithm in details of JPEG files from the view of decoding instead of encoding, dicusses some problems and solutions encountered in practice.

Key words: JPEG Decoding; Image Compression Algorithm; Run Length Encoding; Discrete Cosine Transform; Huffman Encoding