一种基于八叉树的 Huffman 解码方法及其在 MPEG-4 中的应用

汪　林　　殷福亮　　陈　喆
(大连理工大学电子与信息工程学院　大连　116023)

摘　要：传统的二值Huffman解码方法的解码效率较低。为了提高解码速度，该文提出了一种基于八叉树的Huffman解码方法。该方法将 Huffman 码表示为八叉树结构，并根据各个节点在树中的位置将码表重建为一维数组。解码时，每次从码流中读取 3 bit 码元，并使用数值计算代替判断和跳转操作，从而提高了解码效率。将本文方法应用于 MPEG-4 VLC 和 RVLC 解码的实验结果表明，该方法在内存增加不大的情况下能大幅度提高 Huffman 解码效率，其性能优于其它方法。

关键词：数据压缩；Huffman 解码；八叉树；MPEG-4
中图分类号：TN911.72　　**文献标识码**：A　　**文章编号**：1009-5896(2008)08-1861-04

An Octonary Huffman Decoding Method and Its Application to MPEG-4

Wang Lin　　Yin Fu-liang　　Chen Zhe
(*School of Electronic and Information Engineering*, *Dalian University of Technology*, *Dalian* 116023, *China*)

Abstract: The efficiency of traditional binary Huffman decoding method is very low. A new decoding method based on octonary Huffman trees is presented in this paper to improve the decoding speed. Huffman codes are represented as an octonary tree and reconstructed as a single dimensional array according to the position of each node in the tree. When decoding, three bit code elements are read from the bitstream each time, and direct numerical computation may be used to replace "judge and jump" operations, which improve decoding efficiency. The proposed method is also applied to VLC and RVLC decoding algorithms of MPEG-4. Experiment results demonstrate that the proposed method can greatly improve decoding efficiency without increasing memory consumption much, and it exhibits better performance than other decoding methods.

Key words: Data compression; Huffman decoding; Octonary tree; MPEG-4

1　引言

Huffman 编码是由 Huffman 于 1952 年提出的一种无损信源编码方法[1]，它能够无误差地用最少比特数表示信源输出，恢复信源内容。由于 Huffman 码同游程编码相结合可以提供很高的压缩比，因此 Huffman 码是数据压缩中的一种重要方法，在文本、音频、图像、视频等各种类型的数据压缩中有着广泛的应用[2, 3]。Huffman 编解码算法的效率，对这些数据压缩系统的运行效率有很大影响。

在 Huffman 编码方案中，通常要为出现概率较小的信源分配较长的码字，而对出现可能性较大的信源分配较短的码字，以保证产生的编码具有平均最小长度，因此 Huffman 编码算法比较复杂[1−4]。但在实际工程应用中，通常会在编码端提供一个码表，编码时通过查表方法得到所需的码字。在解码端，由于 Huffman 码的长度不固定而且在码流中没有标志位将各个码字隔开，因此解码器必须同时解出每个码字的码长和对应的字符，才能将解码进行下去。Huffman 解码一般通过在码流中逐比特搜索的方法来实现，由于码长未知，因此每读入一个比特后，都要做“判断并跳转”的操作，当码长较大时，解码的效率很低。如何提高 Huffman 解码的效率，已经成为一些音频、视频解码系统能否实时实现的瓶颈之一。

为了提高 Huffman 解码的效率，人们已经提出了多种方法，如二值树法，查表法等[5−9]。Lee 提出了改进的二值解码方法，用数值计算代替每步解码时的判断跳转操作，但该方法解码速度取决于码长的大小，其运算量与码长成正比[10]。Hashemian 方法每次从码流中读取 r 比特码元，减少了解码效率对码长的依赖，但在每一步解码计算下一分支地址时，需要频繁地查表操作，这对解码效率有一定的影响[11, 12]。Aggarwal 提出的基于特征分类的解码算法，虽然解决了解码效率对码长的依赖问题，但它对码字的形式有特殊的要求，不能用作通用的解码算法[13]。为此，本文将 Hashemian 解码方法和改进的二值解码方法相结合，提出了一种基于八叉树的 Huffman 解码方法，并将该方法应用到视频压缩标准 MPEG-4 的 VLC 和 RVLC 解码算法中，同时与其它解码方法的性能进行了比较。实验结果表明，本文提出的方法在内存消耗增加不大的情况下可大幅度提高 Huffman 解码效率。

2007-01-11 收到，2007-08-13 改回

2 基于八叉树的 Huffman 解码新方法

Huffman 码通常由二值 Huffman 树表示。例如，表 1 中的 Huffman 码表可以用图 1 所示的二值 Huffman 树表示。Huffman 树由 3 类节点组成：根节点、叶节点和中间节点，节点之间由枝相连。根节点表示 Huffman 树的开始；叶节点表示某个码字的结束，在叶节点处存储着码字所代表的字符；中间节点对应码字的码元，码长越大，从根节点到叶节点经过的中间节点就越多。码树的高度由码表的最大码长决定。解码时从根节点出发，根据码流中的码元选择中间节点和路径，最终到达叶节点并得到码字所代表的字符，这就是二值 Huffman 树解码方法。该方法是一种传统的解码方法，具有直观和易于实现的优点，但解码速度较慢，解码时间取决于码长的大小。为了提高 Huffman 解码效率，本文将 Hashemian 解码方法和改进二值解码方法相结合[10−12]，提出了一种基于八叉树的 Huffman 解码方法。该方法将 Huffman 树重建为一维数组，解码时每次从码流中读取 3 bit 的码元，并对码元值直接计算得到下一分支的地址，以减少判断和跳转的操作，从而提高了解码效率。下面对该解码方法作详细论述。

表 1 Huffman 码表

字符	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
码字	00	01	10	1100	1101	1110	11110	11111

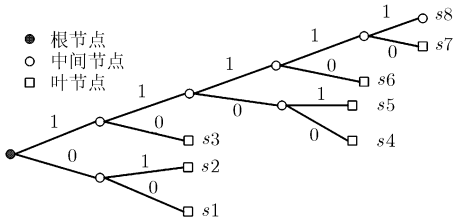

图 1 二值 Huffman 树

传统的二叉树方法解码时每次从码流中读取一个码元，解码效率很低。如果每次从码流中读取 r 个码元，则可以将解码速度提高到原来的 r 倍。基于这个思想，本文将二叉树变形为八叉树，树中每个节点对应 3 个码元，每个节点下面最多有 8 个子节点，分别对应 2^3 种情况：“000~111”。解码时，每经过一个节点，相当于从码流中读取 3 个比特。当码长为 3 的整数倍时，最后一组剩余的码元个数为 3，该组码元与节点是一一对应关系。当码长不是 3 的整数倍时，最后一组剩余的码元个数小于 3，则该组码元对应多个节点，即以该组码元为前缀的码长为 3 的所有节点都同该组码元相对应。设最后一组的有效码元数为 n，则该组码元对应 $2^{(3-n)}$ 个节点。例如“11110”的最后一组码元为“10”，则节点“100～101”中的有效码元都为“10”。由于 Huffman 码为前缀码，码中无任何码字为其它码字的前缀，所以相同前缀的码字都可以解出相同的结果。使用该方法，可以将图 1 中的 Huffman 树变形为八叉树，如图 2 所示。

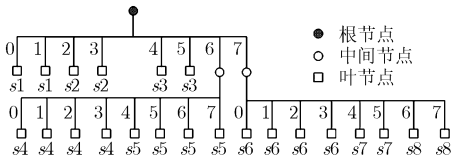

图 2 八叉 Huffman 树

将二叉树变形为八叉树后，将八叉树重建为一维数组，数组中每一元素对应一个节点，供解码时使用。在重建数组时，将八叉树中的节点按照从上到下、从左到右的顺序排列后，依次分配给数组中的元素，数组元素的索引值随着节点数目的增加而增加。每个节点包含 3 个变量：F，ret，cl。其中 cl 表示节点中的有效码元个数；F 表示该节点是否为叶节点，如果 F 为 0，则该节点为中间节点，cl 的值为 3，ret 表示该节点到它的 0 子节点的偏移距离，若 F 为 1，则该节点为叶节点，cl 的取值为 1~3，ret 为码字对应的字符。基于八叉树的重建一维数组 DATA[ ]如表 2 所示。该数组中的每个元素代表一个节点，表中最左边的一列“层”表明了节点在树中所属的层数，表中的前 8 个元素对应第 1 层的节点，后 16 个元素对应第 2 层的节点。例如，第 1 层的中间节点 6 在表中的索引值为 6，它的 ret 值为 2，表明它到它的子节点 0 的偏移距离为 2，即索引值为 8 处的元素。

根据重建数组 DATA[ ]，可以根据读入码流直接运算得到下一分支的地址为

$$\text{index:=index+DATA[index].ret+new\_3digit} \quad (1)$$

解码的流程图如图 3 所示。该方法每次从码流中读取 3 bit 码元，并根据读入的码流值直接计算得到下一分支地址，从而减少了冗余的查表操作，提高了解码效率。

表 2 重建的一维数组

index	层 1								层 2															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
F	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
cl	2	2	2	2	2	2	3	3	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2
ret	S_1	S_1	S_2	S_2	S_3	S_3	2	10	S_4	S_4	S_4	S_4	S_5	S_5	S_5	S_5	S_6	S_6	S_6	S_6	S_7	S_7	S_8	S_8

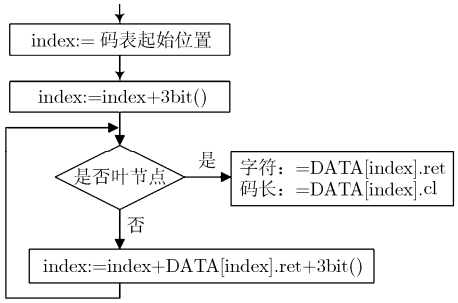

图 3 八叉树 Huffman 解码流程

为说明解码过程，我们以码流“0011110”为例，从码流中读取 3 bit 码元“001”，计算出索引值为 1，对应节点为叶节点，对应的字符为 S_1，有效码长为 2，将码流偏移 2 bit 开始下一码字的解码。读取 3 bit 码元“111”，计算出索引值为 7，对应节点为中间节点，它到下一子节点 0 的偏移距离为 10，计算出子节点 0 索引值为 17，再从码流中读取 3 比特码元“100”或“101”，对应节点的索引值为 21 或 22，这两个节点的内容都一样，都是叶节点，对应的码字为 S_7，有效码长为 1，所以整个码流对应的字符为 S_1S_7。

3 实验结果:八叉树 Huffman 解码方法在 MPEG-4 中的应用

为了验证本文的八叉树解码方法的有效性，我们将该方法应用于视频压缩标准 MPEG-4 的 Huffman 解码算法中[14, 15]。MPEG-4 标准是由国际运动图像专家组于 2000 年提出的一种基于对象的多媒体视频压缩标准，它以高质量、低传输率的特点广泛应用于网络多媒体、视频会议和多媒体监控等系统中。Huffman 编码是 MPEG-4 标准中的一个重要模块，它同游程编码相结合，可以获得近似最优熵编码的性能，提供很高的压缩比。由于编码码流有很大一部分由 Huffman 码构成，所以 Huffman 解码效率关系到整个 MPEG-4 解码器的性能。

MPEG-4 标准中的 Huffman 码表可以分为两类：变长码 (VLC) 和可逆变长码 (RVLC)。VLC 码属于正常的 Huffman 编码，它的码长较小；RVLC 码是带有纠错功能的 Huffman 编码，可以双向解码，但码长较长。为了比较各种 Huffman 解码算法的性能，本文将几种不同的算法应用到 VLC 和 RVLC 解码中。对 VLC 码，分别使用传统的二值解码方法(M1)、改进的二值解码方法(M2)[10]，Hashemian 解码方法(M3)[11, 12]，特征分类解码方法(M5)[13]以及本文提出的解码方法(M4)。对 RVLC 码，由于码表中的码字不具备 M5 中的分类条件，所以只使用 M1，M2，M3 和 M4 这 4 种通用解码方法。在实验中，选取 4 个标准视频测试序列 foreman，clarie，coastguard 和 container，在图像大小为 QCIF，帧率为 15 帧/秒，帧长为 60 帧的条件下，分别使用 VLC 和 RVLC 编出不同码率的 Simple Profile 格式码流作为测试文件。

Huffman 解码方法的计算复杂度由其消耗的总指令数来衡量。在计算指令数时，只记录比较跳转和内存读取的次数，一次比较跳转或内存读取计为一条指令。方法 M5 中的判断比特变换码位置的操作通常在 DSP 有专门的指令完成[16]，所以也计为一条指令。上述 5 种 Huffman 解码方法对测试文件解码的实验结果如表 3 所示，各种解码方法的内存消耗统计如表 4 所示。从表 3 可以看出，本文方法(M4)消耗的指令个数要小于 M1，M2 和 M3 方法，而高于 M5 方法。从表 4 可以看出，本文方法要比其它几种方法消耗更多的内存，但内存消耗的增加量不大，最大不超过 1.4 k byte。

表 3　5 种 Huffman 解码方法性能比较

视频测试序列	码表格式	码率 (kbps)	指令总数(10^6)				
			M1	M2	M3	M4	M5
foreman	VLC	128	2.07	1.20	0.87	0.48	0.32
clarie	VLC	64	1.18	0.73	0.48	0.28	0.18
coastguard	VLC	48	0.93	0.57	0.36	0.20	0.14
container	VLC	48	0.89	0.53	0.35	0.19	0.15
foreman	RVLC	128	1.91	1.11	0.78	0.44	/
clarie	RVLC	64	1.22	0.73	0.46	0.26	/
coastguard	RVLC	48	0.79	0.48	0.33	0.19	/
container	RVLC	48	0.85	0.52	0.33	0.18	/
平均	/	/	1.23	0.73	0.50	0.28	0.20

表 4　5 种解码方法内存消耗(k byte)

	M1	M2	M3	M4	M5
VLC	1.2	0.8	1.1	1.4	1.2
RVLC	2.2	1.4	1.8	2.8	/

为进一步分析解码方法的性能，定义本文方法与其它方法的相对复杂度 RC 为

$$\mathrm{RC}=\frac{\text{待比较方法指令个数}}{\text{本文方法的指令个数}}$$

本文方法相对其它方法的性能改进 PI 为

$$\mathrm{PI}=\left(\frac{1}{\mathrm{RC}}-1\right)\times 100\%$$

由表 3 计算出本文方法同其它方法的相对复杂度比较如图 4 所示。从图 4 可以看出，本文方法(M4)的计算复杂度比其它几种通用方法都要小，平均分别为 M1 的 22%，M2 的 38%，M3 的 58%；其 Huffman 解码性能有了很大提高，相对于 M1 提高了 354%，相对于 M2 提高了 163%，相对于 M3 提高了78%。相对于方法 M5，本文方法的计算复杂度和内存消耗都要更大一些，其计算复杂度平均为 M5 的 140%。这是因为 M5 方法与前几种方法的解码原理不同，解码速度不受码长限制，但该方法通用性较差，对码字的形式有

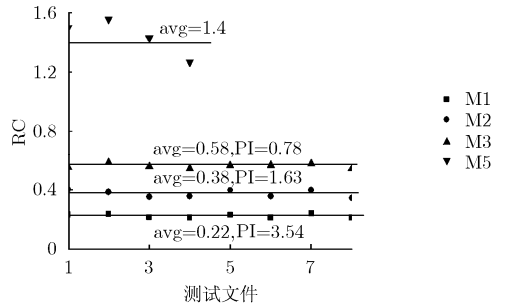

图 4 Huffman 解码方法的相对复杂度比较

一定要求，应用范围较小，例如该方法不能应用于 RVLC 解码。

4 结束语

本文将 Hashemian 方法和改进的二叉树解码方法相结合，提出了一种基于八叉树的 Huffman 解码方法。该方法的主要特点是将 Huffman 码表示为八叉树，并根据节点在树中的位置将码表重建为一维数组，在解码时，用直接的数值运算来代替频繁的查表操作，从而大幅度提高了解码效率。本文将该方法应用于 MPEG-4 的 VLC 和 RVLC 解码算法中，并同其它解码方法进行了比较。实验结果表明，作为一种通用的 Huffman 解码方法，该方法虽然在内存效率方面略有降低，但在解码效率方面有很大提高。此外，根据本文思想，也可以将该方法推广到 $M(M=2^r)$叉树的解码算法中。通常 r 不宜太大，否则随着 r 的增大，占用的内存会明显增加。综合考虑解码速度和内存消耗，我们认为 r 取 3 或 4 比较合适。同时，如何提高本文方法的内存利用率，也是下一步工作的重点。

参 考 文 献

[1] Huffman D A. A method for the construction of minimum redundancy codes. *Proceedings of IRE*, 1952, 40(10): 1098–1101.

[2] Jain A K. Image data compression: An overview. *Proc. IEEE*, 1981, 69(3): 349–389.

[3] Bell T C, Cleary J G, and Witten I H. Text Compression. New Jersey: Prentice-Hall, 1990: 102–108.

[4] Schwartz E S and Kallick B. Generating a canonical prefix encoding. *Communications of the ACM*, 1964, 7(3): 166–169.

[5] Chung K L and Wu J G. Level-compressed Huffman decoding. *IEEE Trans. on Communications*, 1999, 47(10): 1455–1457.

[6] Tanka H. Data structure of Huffman codes and its application to efficient encoding and decoding. *IEEE Transactions on Information Theory*, 1987, 33(1): 154–156.

[7] Ho S and Law P. Efficient hardware decoding method for modified huffman code. *Electronics Letters*, 1991, 27(10): 855–856.

[8] Nekritch Y. Decoding of canonical Huffman codes with look-up tables. Proceedings of Data Compression Conference, Snowbird, UT, USA, 2000: 566–567.

[9] Hashemian R. Condensed table of Huffman coding, a new approach to efficient decoding. *IEEE Trans. on Communications*, 2004, 52(1): 6–8.

[10] Lee J S, Jeong J H, and Chang T G. An efficient method of Huffman decoding for MPEG-2 AAC and its performance analysis. *IEEE Trans. on Speech and Audio Processing*, 2005, 13(6): 1206–1209.

[11] Hashemian R. Design and hardware implementation of a memory efficient Huffman decoding. *IEEE Trans. on Consumer Electronics*, 1994, 40(3): 345–352.

[12] Hashemian R. Memory efficient and high speed search Huffman coding. *IEEE Trans. on Communications*, 1995, 43(10): 2576–2581.

[13] Aggarwal M and Narayan A. Efficient Huffman decoding. International Conference on Image Processing, Vancouver, BC, 2000: 936–939.

[14] International Standard ISO/IEC 14496-2 (MPEG-4). Information Technology–Coding of Audio-Visual Objects–Part 2: Visual. 2002.

[15] ISO/IEC JTC1/SC29/WG11 N4668. Overview of MPEG-4 Standard. 2002.

[16] Texas Instrument. TMS320C6000 CPU and Instruction Set Reference Guide. 1999.

汪 林： 男，1981年生，博士生，研究方向为数字信号处理、音频与视频信号处理.

殷福亮： 男，1962年生，教授，博士生导师，主要研究方向为数字信号处理、语音处理、图像处理、宽带无线通信技术和集成电路设计.

陈 喆： 男，1975 年生，副教授，博士，研究方向为语音处理、阵列信号处理、现代通信技术.