H.264 标准中 CAVLC 解码的子表 – 哈夫曼算法

张 宇
（同济大学 微电子学与固体电子学，上海 201804）

摘 要：在分析 H.264 中 CAVLC 标准解码算法的基础上提出一种改进算法。此种算法利用子表、哈夫曼树及哈夫曼编码理论，解决了 CAVLC 标准解码算法查找效率不高的问题。实验表明采用几个简单的码表并对每个码表采用哈夫曼编码可以提升约 3 倍的效率。

关键词：H.264; CAVLC; 子表; 哈夫曼树; 哈夫曼编码
中图分类号：TN919.81 **文献标识码**：A

The Subtable-Huffman algorithm on CAVLC in H.264

ZHANG Yu
(Microelectronics & Solid Electronics, Tongji University, Shanghai 201804,China)

Abstract: Based on the analysis of standard decoding methods of CAVLC in H.264, this paper proposes an improved algorithm. Depending on the theories of subtable, Huffman Tree and Huffman coding, this algorithm improves the standard table look-up decoding efficiency on CAVLC. The testing result shows the Subtable-Huffman algorithm achieves 3 times higher efficiency than the standard one, with some simple subtables and the use of Huffman coding, and can improve more.

Key words: H.264; CAVLC; Subtable; huffman tree; huffman coding

自从国际电信联盟标准化部门 ITU-T 和国际标准化组织运动图像专家组 MPEG 联合提出 H.264 视频编解码标准后，该标准几乎成为业内的指向标。各专业部门，研究机构以及众多厂商纷纷开始参与技术完善和产品开发。上下文自适应可变长编码 CAVLC 作为标准中的一部分，性能优劣直接影响整个编解码的效率。

本文通过对 JM86 解码模型和子表、哈夫曼树及编码理论的研究，提出一种 CAVLC 解码优化算法，提升解码器效率，通过实验模拟验证。

1 H.264 中标准 CAVLC 解码算法

1.1 H.264 中标准 CAVLC 解码算法流程

在 CAVLC 解码的过程中，首先要解码出非零系数的数目（TotalCoeffs）和拖尾系数的数目（TrailingOnes），然后解码出拖尾系数的符号。接下来解码除了拖尾系数之外的非零系数的幅值（Levels），再解码出最后一种非零系数前零的数目（TotalZeros），最后解码出每个非零系数前零的个数（RunBefore），这样就确定了（Level，Run）的组对值，也就知道了经过变换、量化后的残差值[1]。

1.2 H.264 种标准 CAVLC 解码算法分析

CAVLC 解码算法是通过查找变长表格从而求出 TotalCoeffs 和 TrailingOnes。该变长表格的存储结构为二维结构，存储的内容为查找的码字，而所对应的二维结构的下标分别代表 TotalCoeffs 和 TrailingOnes。而所要查找的表格由 Nc 确定，查找表包括 3 个变长码表和 1 个定长码表。Nc 的值是根据当前块左边 4 × 4 块的非零系数数目 Na 和当前块上边 4 × 4 块的非零系数数目 Nb 得到，即 Nc=（Na+Nb）/2。这体现了基于上下文的思想。

根据实验得知，在标准 CAVLC 解码过程中，求得 TotalCoeffs 和 TrailingOnes 的时间占总消耗时间的 75%。求 TotalZeros、RunBefore、Levels 的时间分别占 15.4%、6.2%、1.6%，其他占 1.8%。虽然在查找变长码表的过程中考虑了信源符号出现的概率，但解码是对码字依次匹配的命中率仍然不足 40%[2]。所以提高变长码表的查找效率对提升整个 CAVLC 解码效率起着举足轻重的作用。

2 CAVLC 子表 – 哈夫曼算法

2.1 CAVLC 子表 – 哈夫曼算法的原理[3]

根据上面描述，提高变长码表查找效率可以大幅度

地提升整个解码效率。子表是通过分解各个查找码表，缩小查找范围，直接减少搜索查找时间。而哈夫曼树作为最优二叉树可以最优查找路径，基于哈夫曼树的哈夫曼编码可以进一步优化查找码表，间接减少查找时间。而这两种方式的结合可以提升效率空间，如表1所示。

表1 CAVLC各种查找方式比较次数比较[4]

	顺序查找	哈夫曼	子表	子表-哈夫曼
最大比较次数	62	16	24	8.3
平均比较次数	31.5	5.1	10.75	6

2.2 CAVLC 子表－哈夫曼算法的实现

CAVLC子表－哈夫曼算法建立在哈夫曼编码理论之上，所有编码都是前缀编码，即任意编码都不是其他编码的前缀，码字的长度不是一定的，并且码长较短的码字出现概率比较大。

以0≤Nc<2的情况为例。首先需要采用子表方式初步缩小查找的范围。将码字按位分组，分组的原则为每4 bit为一组，每次读入一组，按照每组性质的不同来进行子表的划分。具体划分方法为：第1个码表存放第一组（第一个4 bit）不为“0000”的码字；第2个码表存放第一组为“0000”且第二组（第二个4 bit）不为“0000”的码字；第三个码表存放第一、二组为“0000 0000”且第三组（第三个4bit）不为“0000”的码字；第四个码表存放剩余的全部码字。

第二步是要根据哈夫曼树及哈夫曼编码理论重新构造码表的查找搜索结构，也就是构建每个表格的哈夫曼树，使该表的查找效率最优。具体构造方法为：为树中每个节点的左，右分支节点分别用0和1编码，而每个叶节点的码字就是从根节点到该叶节点所经过的路径上的0和1编码构成的编码序列。这个码字就是所要查找的内容，而该码字所对应的叶节点的值就是该码字在重新构造的变长码表中的坐标（TrailingOnes，TotalCoeffs）。而该哈夫曼树的左右非叶节点的值统一为（-1，-1）。

使用以上方法得到了全新的变长码表组，这种构造变长码表组，并利用该码表组进行查找的方法就是子表－哈夫曼算法[5]。具体的查找方法如下：

(1)确定所要查找的变长码表。从码流中读取第一个4 bit，若其值为“0000”则继续查找第二个4 bit，否则查找码表1。若第二个4 bit的值为“0000”则继续查找第三个4 bit，否则查找码表2。若第三个4 bit的值为“0000”则查找码表4，否则查找码表3。最后设置当前节点为根节点。

(2)读码流中的第一位赋给变量bit；

(3)if（bit为0且该节点有左分支节点）

{移动当前节点指针到左分支节点}；

elseif（bit为1且该节点有右分支节点）

{移动当前节点指针到右分支节点}；

Else{转至步骤(5)}；

(4) 跳转回步骤(2)；

(5) 若当前节点的值不为（-1，-1），则查找成功，该节点的值就是所要查找的（TrailingOnes，TotalCoeffs）。否则查找失败。

分解后的各个码表如表2所示。

表2 子表-哈夫曼算法分解变长码表

TotalCoeffs	TrailingOnes 0	1	2	3
0	1			
1	0001 01	01		
2	0000 0111	0001 00	001	
3	0000 0011 1	0000 0110	0000 101	0001 1
4	0000 0001 11	0000 0011 0	0000 0101	0000 11
5	0000 0000 111	0000 0001 10	0000 0010 1	0000 100
6	0000 0000 0111 1	0000 0000 110	0000 0001 01	0000 0100
7	0000 0000 0101 1	0000 0000 0111 0	0000 0000 101	0000 0010 0
8	0000 0000 0100 0	0000 0000 0101 0	0000 0000 0110 1	0000 0001 00
9	0000 0000 0011 11	0000 0000 0011 10	0000 0000 0100 1	0000 0000 100
10	0000 0000 0010 11	0000 0000 0010 10	0000 0000 0011 01	0000 0000 0110 0
11	0000 0000 0001 111	0000 0000 0001 110	0000 0000 0010 01	0000 0000 0011 00
12	0000 0000 0001 011	0000 0000 0001 010	0000 0000 0001 101	0000 0000 0010 00
13	0000 0000 0000 1111	0000 0000 0000 001	0000 0000 0001 001	0000 0000 0001 100
14	0000 0000 0000 1011	0000 0000 0000 1110	0000 0000 0000 1101	0000 0000 0001 000
15	0000 0000 0000 0111	0000 0000 0000 1010	0000 0000 0000 1001	0000 0000 0000 1100
16	0000 0000 0000 0100	0000 0000 0000 0110	0000 0000 0000 0101	0000 0000 0000 1000

备注：码表1　码表2　码表3　码表4

以码表 1 为例，构建哈夫曼树（图 1）。

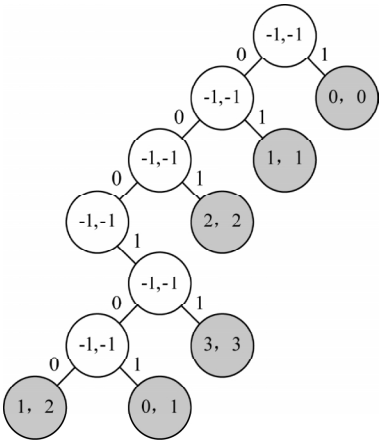

图 1 码表 1 构建哈夫曼树

3 仿真结果

以 JM86 为实验模型，选取 3 张样图作为实验对象，测试解码查表时间。首先，直接测试 JM86 解码的查找时间，之后采用子表－哈夫曼算法建立变长码表组，重新测试解码的查找时间，结果如图 2 所示。图 3 为优化算法前后节省时间的比例。

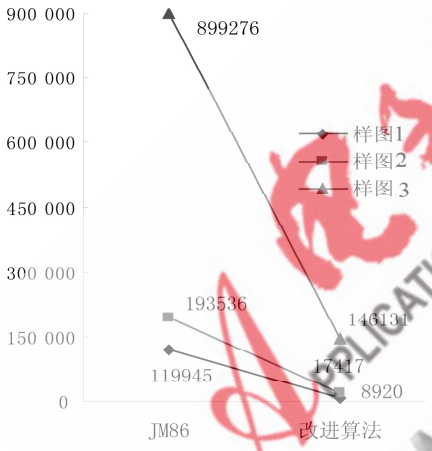

图 2 算法改进前后运行时间比较图

从实验结果可以看出，采用子表－哈夫曼算法优化查找码表之后节省了 90% 左右的时间，即改进算法之后的查找运行时间仅为改进之前的 10% 左右。根据之前的实验理论，由（75% × 10%+25%）/（75%+25%）=32.5% 可知，通过采用子表－哈夫曼算法改进查找码表的方法使整个 CAVLC 的运行时间缩短为改进之前的 32.5%。即整体效率提升至原来的 3 倍左右，并且随着分组的复杂化，仍存在很大的提升空间。可见子表－哈夫曼算法对于 CAVLC 解码的优化是十分有效的。

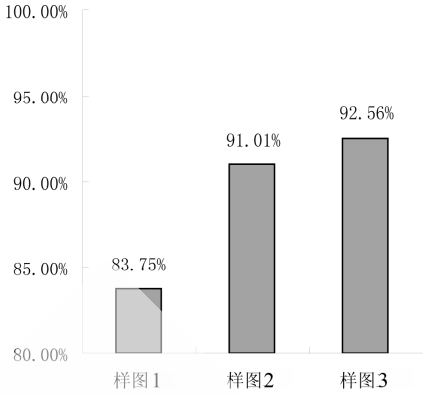

图 3 算法改进前后样图时间减少比例

参考文献

[1] 毕厚杰.新一代视频压缩编码标准:H.264/AVC[M].北京:人民邮电出版社,2005.

[2] KIM D K, DO S H, CHO H. A new joint algorithm of symbol timing recovery and sampling clock adjustment for OFDM systems [J]. IEEE Trans. on Consumer Electronics, 1998, 44(3):1142- 1149.

[3] 严蔚敏，吴伟民. 数据结构(C语言版).北京：清华大学出版社，1997.

[4] 童伟，支琦，宋利，等. H.264中CAVLC解码的高校算法[J]. 微计算机信息，2006，22(9-2).

[5] 张银芬.基于H.264的解码算法优化[D].西安电子科技大学，2007.

（收稿日期：2009-04-16）