第37卷 第15期 计 算 机 工 程 2011年8月
Vol.37 No.15 Computer Engineering August 2011

·多媒体技术及应用· 文章编号：1000—3428(2011)15—0218—03 文献标识码：A 中图分类号：TP311

一种新的AVS指数哥伦布码算法

文 斌，何明华，黄敏琪

(福州大学电气工程与自动化学院，福州 350108)

摘 要：指数哥伦布码是AVS视频压缩标准中熵编码的重要组成部分。在研究指数哥伦布码的特点及其编解码算法的基础上，利用哥伦布码码字的二进制大小与编码码值的数学关系，提出一种新的指数哥伦布编解码算法。尤其是在指数哥伦布解码方面，摒弃变长码只能逐位读取、逐位判断计算的思路，采用32 bit读取并用设定公式计算的方法。实验结果表明，该算法比AVS参考代码中的指数哥伦布编、解码方法所用时间分别缩短约10%和30%。

关键词：AVS标准；指数哥伦布码；熵编码；视频编码

AVS Exponential-Golomb Code Algorithm

WEN Bin, HE Ming-hua, HUANG Min-qi

(School of Electrical Engineering & Automation, Fuzhou University, Fuzhou 350108, China)

【Abstract】The exponential-Golomb code is a important part of entropy encoding in AVS video compression standard. This paper investigates the characteristics, encoding and decoding algorithm of exponential-Golomb code, and presents a new algorithm for Exponential-Golomb encoding and decoding according to the mathematic relation between binary value of exponential-Golomb code. Especially in exponential-Golomb decoding, the algorithm discards the idea of reading and judgement bit-by-bit, uses the method of 32 bit reading and given formula. Experimental result shows that the exponential-Golomb code algorithm shortens the encoding time by about 10% than the algorithm applied in AVS reference software, and shortens the decoding time by about 30% than the algorithm applied in AVS reference software.

【Key words】AVS standard; exponential-Golomb code; entropy coding; video coding

DOI: 10.3969/j.issn.1000-3428.2011.15.070

1 概述

AVS(Advanced audio Video coding Standard)是由国家信息产业部科学技术司于2002年6月批准成立的数字音视频编解码技术标准工作组所制定的标准。它包括系统、视频、音频、一致性测试、参考软件、数组媒体版权管理、移动视频、在IP网络上传输AVS、AVS文件格式9个部分[1]。AVS视频编解码标准采用一系列技术来达到高效率的视频编码，包括帧内预测、帧间预测、变换、量化和熵编码[2]等。在AVS视频编解码标准中，所有的语法元素都是基于指数哥伦布码或者定长码设计的。定长码用来编码具有均匀分布的语法元素，指数哥伦布码用来编码可变概率分布的语法元素。指数哥伦布码的编解码效率直接影响到AVS视频编解码的效率。

本文在研究指数哥伦布码的数学特性和AVS参考代码中的指数哥伦布编解码算法的基础上，利用哥伦布码字的二进制大小与编码码值的数学关系，提出了新的指数哥伦布编解码方法。

2 指数哥伦布码

指数哥伦布码[3]的优势在于：一方面，它的硬件复杂度比较低，可以根据闭合公式解析码字，无需查表；另一方面，它可以根据编码元素的概率分布灵活地确定以k阶指数哥伦布码编码，如果k选得恰当，则编码效率可以逼近信息熵。

表1给出了0阶和1阶指数哥伦布码的结构。指数哥伦布码的比特串分为“前缀”和“后缀”两部分[1]。前缀由*leadingZeroBits*个连续的“0”和1个“1”构成。后缀由*leadingZeroBits* +k个比特构成，即表中的x_i串，x_i的值为“0”或“1”。

表1 k阶指数哥伦布码表

阶数	码字结构	CodeNum取值范围
k=0	1	0
	0 1 x_0	1~2
	0 0 1 x_1 x_0	3~6
	…	…
k=1	1 x_0	0~1
	0 1 x_1 x_0	2~5
	0 0 1 x_2 x_1 x_0	6~13
	…	…

2.1 指数哥伦布码解码

在解析k阶指数哥伦布码时，首先从比特流的当前位置开始寻找第1个非零比特，并将找到的零比特个数记为*leadingZeroBits*，然后根据*leadingZeroBits*计算*CodeNum*。用伪代码描述如下：

```
leadingZeroBits =－1;
for ( b = 0; ! b; leadingZeroBits++ )
b = read_bits(1);
CodeNum = 2^(leadingZeroBits + k) – 2^k + read_ bits(leading ZeroBits + k);
```

其中，*CodeNum*为解码得到的码值；*read_bits*(n)函数返回比特流的随后n个比特。AVS参考代码中的哥伦布解码就是按照该算法实现的。

2.2 指数哥伦布码编码

将码值*CodeNum*进行编码，指数哥伦布码码字结构

基金项目：福建省自然科学基金资助重点项目(2007J0003)；福建省新世纪优秀人才支持计划基金资助项目(XSJRC2007-26)

作者简介：文 斌(1985－)，男，硕士研究生，主研方向：指数哥伦布码算法，嵌入式系统；何明华，教授、博士；黄敏琪，硕士研究生

收稿日期：2011-01-25 **E-mail：**wenbin_08@126.com

如下：

$$\left\{\begin{array}{l} leadingZeroBits\text{个}0 \\ \end{array}\right\}[1]\left\{\begin{array}{l} leadingZeroBits+k \\ \text{比特的}INFO \end{array}\right\}$$

零比特个数 *leadingZeroBits* 和后缀 *INFO* 的计算公式如下：

$$leadingZeroBits = 1\mathrm{b}(CodeNum/2^k+1) \quad (1)$$

$$INFO = CodeNum-(2^{leadingZeroBits}-1)\times 2^k \quad (2)$$

AVS 参考代码中的指数哥伦布编码并不是直接用上述公式计算，而是用一系列循环移位和加减操作来得到所要的码字，AVS 参考代码中的指数哥伦布编码方法的伪代码描述如下：

```
res =(1<<k );
numbits=1+k;
Value = CodeNum;
While (Value >= res)
        {       Value－= res;
                res << = 1 ;
                numbits+=2;
         }
codebits=res|Value;
codelen=numbits;
```

其中，*k* 为阶数；*CodeNum* 为编码码值；*res*、*numbits* 为临时变量；*Value* 的值表示后缀大小；*codebits* 为编码码字；*codelen* 为哥伦布码码长。

3　一种新的指数哥伦布码算法

在解析 *k* 阶指数哥伦布码时，首先找到零比特个数并记为 *leadingZeroBits*，然后根据 *leadingZeroBits* 计算 *CodeNum*。计算式如下：

$$CodeNum = 2^{leadingZeroBits+k}-2^k+read\_bits(leadingZeroBits+k) \quad (3)$$

被编码的哥伦布码字的二进制大小 *GolombValue*(哥伦布码字的前缀 1 和后缀组成的比特串的值)与 *leadingZeroBits* 及阶数 *k* 有如下关系：

$$GolombValue = 2^{leadingZeroBits+k}+read\_bits(leadingZeroBits+k) \quad (4)$$

由式(3)变形得：

$$CodeNum+2^k = 2^{leadingZeroBits+k}+read\_bits(leadingZeroBits+k) \quad (5)$$

由式(4)、式(5)得：

$$CodeNum+2^k = GolombValue \quad (6)$$

即哥伦布码字的二进制大小 *GolombValue* 为编码码值 *CodeNum* 与 2^k 的和。

3.1　本文指数哥伦布码编码算法

根据式(6)，得到一个新的哥伦布编码方法如下：

(1)确定要编码的码值 *CodeNum* 和阶数 *k*。

(2)求出 *CodeNum* 与(1<<*k*)的和，赋值给哥伦布码字的二进制大小 *GolombValue*。

(3)根据 *GolombValue* 的二进制码可得到哥伦布码字前缀中连续为 0 的个数 *leadingZeroBits*。

(4)将所述二进制码前加上 *leadingZeroBits* 个连续的 0，得到哥伦布编码码字。

现在以编码 0 阶指数哥伦布码码值 14 举例说明：

(1)编码码值 *CodeNum* 为 14，*k* 为 0。

(2)*GolombValue* 为 14+1=15，二进制表示为 1111。

(3)*leadingZeroBits* 为 4－1=3。

(4)得到哥伦布编码码字为 0001111。

用伪代码表示为：

```
GolombValue=(1<<k)+ CodeNum;
 nn=GolombValue;
for (i=0;    nn != 0; i++)
   {
       nn >>=1;
   }
Len=2×i－1－k ;
```

其中，*k* 为阶数；*CodeNum* 为编码码值；*nn*、*i* 为临时变量；*Len* 为哥伦布码长度；*GolombValue* 是哥伦布码字的二进制大小，也就是码字。

3.2　本文指数哥伦布码解码算法

根据式(6)，得到一种新的指数哥伦布解码方法，步骤如下：

(1)从比特流的当前位置开始寻找第 1 个非零比特，并将找到的零比特个数记为 *leadingZeroBits*。

(2)将找到的第 1 个非零比特及其后面 *leadingZeroBits*+*k* 位赋值给 *GolombValue*。

(3)码值 $CodeNum = GolombValue-2^k$。

现在以解码 0 阶指数哥伦布码 14 的码字 0001111 举例说明：

(1)*leadingZeroBits* 为 3。

(2)*GolombValue* 为 1111，十进制值为 15。

(3)码值 *CodeNum*= 15-1=14。

哥伦布码码长由其码字内容决定，需要逐位读出码字来决定码字的值。这样的逐位读取码字的操作会引入过多的函数调用开销，不利于编译器对程序的优化和容易打断处理器的流水线操作[4]。

本文在上述解码步骤的基础上，抛弃了变长码只能逐位读取、逐位判断计算的思路，采用一次性读取多个比特[5]，并直接对其进行分析解码的办法。AVS 码流中指数哥伦布码字不会超过 32 bit 长，所以，一次读入 32 bit 存放在缓冲区里面，对 32 bit 的缓冲区内容进行计算解码。

假设 32 bit 缓冲区 *buffer* 中的内容如下：

0000 1011 1000 0010 0000 0010 1000 0010

首先要计算出前导的“0”的个数 *leadingZeroBits*。*leadingZeroBits* 的大小如式(7)：

$$leadingZeroBits = 32-(floor(1\mathrm{b}\ buffer)+1) \quad (7)$$

注意到这里的 lb 的作用就是找出非零的最高位的位置，本文直接用移位的方法得到 *leadingZeroBits*，计算量少于直接用式(7)计算的计算量。

然后计算 *GolombValue* 的值。因为 *leadingZeroBits* 的值已经计算出来，只要把 *buffer* 向右移位 *rightshiftBits*，剩下 2*leadingZeroBits*+1+*k* 位的码字，得到 *GolombValue*，即：

$$GolombValue = buffer >> rightshiftBits \quad (8)$$

右移的位数，计算方法如下：

$$rightshiftBits = 32-2\times leadingZeroBits-1-k \quad (9)$$

最后得到码值：

$$CodeNum = GolombValue-2^k \quad (10)$$

因此，本文的指数哥伦布码解码算法的伪代码如下：

```
ctr_bit = 0; len=0; bitLeft=31;
while (ctr_bit==0)
  { ctr_bit= buffer & (0x01<<(bitLeft —));
len++;
  }
rightshiftBits=33 –2×len –k;
```

```
GolombValue= buffer>>rightshiftBits ;
CodeNum=GolombValue – (1<<k) ;
```

其中，*k* 为阶数；*CodeNum* 为哥伦布码值；*len* 为哥伦布码前缀长度；*GolombValue* 是哥伦布码字的前缀 1 和后缀连续二进制字符串码值；*buffer* 是作为缓冲区的 32 bit 的变量，*bitLeft*、*ctr_bit* 为临时变量；*rightshiftBits* 为右移的位数。

4 实验结果及其分析

在保证正确进行指数哥伦布编解码的条件下，本文进行了编解码效率的测试，以 AVS-P2 的参考软件 RM52j 为仿真平台，分别进行编码和解码测试。

编码测试条件包括帧率为 30 Hz，编码 58 帧，帧间预测最多使用 2 个参考帧，采用 IBBP 序列，使用 RD 优化，不使用码率控制，QP 值取 10。考虑到图像大小不影响哥伦布编解码的效率，本文仅选择 CIF 大小的序列进行测试。选择 Highway、Container、Akiyo、Foreman、Hall、Coastguard 6 个 YUV 序列做测试。测试微机配置为 Intel(R)core2 Duo E8500@3.16 GHz CPU，2 GB 内存，Windows XP Professional 操作系统。编码测试结果如表 2 所示。

表 2 哥伦布码编码测试结果

格式	测试 YUV 序列	哥伦布编码时间/ms		时间缩短/(%)
		AVS 参考代码中方法	本文算法	
352×288	Highway	5 204.41	4 573.82	12.12
352×288	Container	5 371.21	4 803.07	10.58
352×288	Akiyo	4 056.58	3 393.43	16.35
352×288	Foreman	4 924.87	4 370.06	11.27
352×288	Hall	5 335.23	4 703.35	11.84
352×288	Coastguard	5 723.71	5 153.06	9.97

解码测试条件为用参考软件 RM52j 的原代码编码 Highway、Container、Akiyo、Foreman、Hall、Coastguard 6 个 YUV 序列，得到各 181 帧的 6 个 AVS 序列 Highway.avs、Container.avs、Akiyo.avs、Foreman.avs、Foreman.avs、Hall.avs、Coastguard.avs 用于解码测试。编码条件与编码测试中的一样。测试微机配置为 Intel(R)Pentium Dual E2140@1.60 GHz CPU，512 MB 内存，Windows XP Professional 操作系统。测试结果如表 3 所示。

表 3 哥伦布码解码测试结果

格式	测试 AVS 序列	哥伦布解码时间/ms		时间缩短/(%)
		AVS 参考代码中方法	本文算法	
352×288	Highway.avs	1 171.64	822.57	29.79
352×288	Container.avs	652.79	454.44	30.38
352×288	Akiyo.avs	171.42	110.12	35.76
352×288	Foreman.avs	883.20	612.16	30.69
352×288	Hall.avs	935.81	623.25	33.39
352×288	Coastguard.avs	1 047.45	675.87	35.47

由表 2 可以看出，本文提出的指数哥伦布编码算法比 AVS 参考代码中的指数哥伦布编码方法时间缩短约 10%。由表 3 可以看出，本文提出的指数哥伦布解码算法比 AVS 参考代码中的指数哥伦布解码方法时间缩短约 30%。

5 结束语

本文在研究指数哥伦布码的数学特性和 AVS 参考代码中的指数哥伦布编解码算法的基础上，提出了一种新的指数哥伦布编解码算法。在保证正确进行指数哥伦布编解码的条件下进行了编解码效率的测试，结果表明，本文提出的指数哥伦布编码算法比 AVS 参考代码中的指数哥伦布编解码方法所用时间都有所缩短。

参考文献

[1] AVS 数字音视频编解码技术标准工作组. GB/T 20090. 2-2006 信息技术先进音视频编码第二部分: 视频[S]. 北京: 中国标准出版社, 2006.

[2] 侯金亭, 马思伟, 高 文. AVS 标准综述[J]. 计算机工程, 2009, 35(8): 247-249.

[3] Wang Qiang, Zhao Debin, Gao Wen. Context-based 2D-VLC Entropy Coder in AVS Video Coding Standard[J]. Journal of Computer Science & Technology, 2006, 21(3): 315-322.

[4] 翁慈洁, 张悠慧, 汪东升. H.264 中指数哥伦布算法的优化实现研究[J]. 计算机工程与设计, 2007, 28(12): 2867-2869.

[5] 虞新阳, 吴成柯, 赵 波. 指数哥伦布码的快速平稳解码算法[J]. 数字电视与数字视频, 2006, 22(2): 14-15.

编辑 索书志

(上接第 207 页)

4 结束语

本文提出了一种验证码倾斜自动校正算法。通过该算法，可以对发生倾斜的数字字符进行自动矫正，使字符最大程度接近无偏转的模板数据，从而提高其识别率，对于特殊的字符，甚至可以直接识别。本文通过对 2 种标准数字字符形式的研究，发现只要符合数字字符的基本特征，一定程度的弯曲和变化并不影响算法的通用性。因此，对于接近标准字符的数字验证码而言，也可以正确找到字符底边并进行倾斜矫正。由于该算法限定的范围为 180°，且对于字符预处理要求较高，因此更好的改进算法有待进一步的研究。但该方法无论对于矫正标准数字还是有一定形变的具有较小噪声的验证码数字都有较好的适应性，稍作改动还可用于字母类型，且矫正后的字符最大程度接近无倾斜的标准字符，这对于后期的识别工作非常有利，能使存储的模板数据量降到最低。出于对数字验证码安全性的考虑，认为设计一种难于分割，包含能对字符结构造成破坏性影响，但又不影响人工识别的噪声的验证码是提高数字验证码安全性的有效方法。

参考文献

[1] Hew P C. Geometric and Zernike Moments[D]. Perth, Australia: The University of Western Australia, 1996.

[2] 芮 挺, 沈春林, 张金林. 车牌识别中倾斜牌照的快速矫正算法[J]. 计算机工程, 2004, 30(13): 122-124.

[3] Madhvanath S, Govindaraju V. The Role of Holistic Paradigms in Handwritten Word Recognition[J]. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2001, 23(2): 149-164.

[4] 蓝章礼. 基于中心与圆周的英文字符识别方法研究[J]. 计算机科学, 2007, 34(4): 241-249.

[5] 袁保社, 吾守尔·斯拉木. 一种手写维吾尔文字母识别算法[J]. 计算机工程, 2010, 36(2): 186-188.

编辑 顾逸斐