基于概率估计更新的 CABAC 加速算法

郭 斌[1),2)] 王维东[1),2)] 叶青青[3)] 沈渝力[1),2)] 章 纮[1),2)]

[1)](浙江大学信电系,杭州 310027) [2)](浙江省综合信息网技术重点实验室,杭州 310027)

[3)](杭州职业技术学院,杭州 310018)

摘 要 基于上下文的自适应二进制算术编码(context-based adaptive binary coding,CABAC)是一种高效的熵编码方法,但是其高计算复杂度制约了该算法的编码速度,已成为其应用的一个主要瓶颈。为解决此问题,在分析 CABAC 算法及其计算复杂度的基础上,对其概率估计更新部分进行了改进,提出了一种提高其编码速度的有效方法。该方法首先以 N 个符号的包为单位进行编码,仅每个包编码完成后再进行一次概率估计的更新,因而成倍地降低了概率估计更新的频度。实验数据表明,该方法较以往的方法使 CABAC 的编码速度平均提高了 13.3% ~ 30.7%,同时编码效率平均下降 1.87% ~2.98% 。

关键词 基于上下文的自适应二进制算术编码 计算复杂度 概率估计更新 编码速度

中图法分类号: TN919.81 **文献标识码**: A **文章编号**: 1006-8961(2009) 02-0281-05

A CABAC Accelerating Algorithm Base on Probability Estimation Update

GUO Bin[1),2)], WANG Wei-dong[1),2)], YE Qing-qing[3)],
SHEN Yu-li[1),2)], ZHANG Hong[1),2)]

[1)] (*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou* 310027)

[2)] (*Zhejiang Provincial Key Laboratory of Information Network Technology, Hangzhou* 310027)

[3)] (*Hangzhou Vocational and Technical College, Hangzhou* 310018)

Abstract Context-based adaptive binary arithmetic coding (CABAC) is a highly efficient entropy coding way, but its coding speed is restricted by high computational complexity, which becomes a major bottleneck in its application. To solve this problem, an effective way, based on the analysis of the CABAC algorithm and its computational complexity, will be proposed in this article to improve the algorithm's coding speed by improving the probability estimation update part. The algorithm first codes packets which each include N symbols, and then updates the probability estimation part, which decreases the update frequency of probability estimation remarkably. The experimental results show that, compared with the former algorithms, the coding speed of CABAC has been substantially increased from 13.3% to 30.7%, with coding efficiency declines a little from 1.87% to 2.98%.

Keywords CABAC, computational complexity, probability estimation update, coding speed

1 引 言

熵编码是一类利用数据的统计信息进行数据压缩的无损编码方法,而基于上下文的自适应二进制算术编码(CABAC)则是一种效率非常高的熵编码方法,它被最新的视频编码标准 H.264[1] 所采用,已在提高视频编码效率方面发挥了重要作用。

基金项目: 浙江省科学技术厅重点科研项目(2004C21052)

收稿日期:2007-04-11;**改回日期**:2007-06-06

第一作者简介:郭 斌(1981 ~),男。浙江大学信息与通信工程专业硕士研究生。主要研究方向为视频压缩技术。E-mail:gbbyron@yahoo.com.cn

通讯作者:王维东。E-mail:wdwang@mail.hz.zj.cn

与广泛使用的通用可变长编码(UVLC)相比,CABAC 充分利用了算术码的特点和视频流码间的相关统计特性,使得编码效率较 UVLC 有了很大的提高。Marpe 等人的实验结果表明,CABAC 能在各种不同码率的情况下,较 UVLC 节省 9% ~14% 的码率[2]。但在提高编码效率的同时,CABAC 要比 UVLC 多大约 40% 的运算量[3],这也制约了 CABAC 的应用范围。

作为一种性能优异的熵编码方法,CABAC 受到了学术界的重视,并对它进行了很多的优化。优化的思路主要集中在以下两方面:一方面是进一步提高编码效率,以降低码率;另一方面是降低编码计算复杂度,以提高编码速度。现有的对 CABAC 进行改进的途径有:(1) 处理器的优化,使得处理器的结构适应 CABAC 编码的特点,以利于流水操作或者一个周期处理多条指令[4];(2) 算法的优化,比如优化概率模型预测的方式和通过提高上下文模型预测的准确性,以提高 CABAC 的编码效率[5]等。由于处理器的优化主要针对特定的处理器硬件设计场合,因此算法的优化更加通用灵活,代价也相对较小。

通过分析 CABAC 编码流程和它的计算复杂度,尤其是编码过程中开销较大的概率估计更新部分,本文提出了一种 CABAC 的加速算法。该算法试图在编码效率下降较少的前提下,有效提升编码速度。

2 CABAC 算法的改进思路

算术编码的基本思想是应用信源序列的累积概率来分割[0,1)区间中互不重叠的子区间,各个子区间的宽度代表相应信源序列出现的概率,可通过选择子区间中的点来代表与该子区间对应的信源序列,该点小数值的二进制代码就是该信源序列的算术码。自适应二进制算术编码需要对 0 和 1 的概率不断地进行更新,以便使概率能够实时准确地反映 0 和 1 的分布。

下面以 H.264 中的 CABAC 应用为例,对其编码流程进行分析。

如图 1 所示,在 H.264 中,CABAC 熵编码过程主要分成上下文建模、二进制化和自适应二进制算术编码 3 个步骤,其中最后一部分由概率估计和编码器组成。

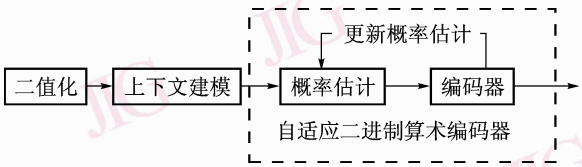

图 1 CABAC 基本框图

Fig. 1 Generic block diagram of CABAC

上下文建模就是利用相邻的已编码符号所提供的相关性,为所要编码的符号选择合适的上下文模型,这个模型提供了对当前待编码符号的条件概率估计。利用合适的上下文模型就可以大大降低符号间的冗余度。

二进制化是 CABAC 中较为独立的一步,它是把非二进制符号映射成若干比特二进制串,以便为后面进行二进制算术编码做好准备。

该 CABAC 编码器的输入值是上下文变量和待编码的二进制值 *binVal*,编码器的状态是当前编码区间 R 以及该区间的下限 L。编码器负责完成区间细分功能。在编码器中,可通过查表的方法完成快速计算,即首先把编码区间量化,因为 $2^8 \leqslant R \leqslant 2^9$,所以可以通过 $R(b_8b_7\cdots b_2b_1b_0)$ 的 b_7b_6 对其进行 4 等分,在编码器中,量化区间的索引值 ρ 可由下式计算得到:

$$\rho = (R >> 6) \& 3 \tag{1}$$

4 个量化区间值分别为 Q_0、Q_1、Q_2、Q_3。在该 CABAC 编码器中,是用 64 个有代表性的概率值来表示最小概率符号(least probable symbol, LPS)的概率。这 64 个概率值可通过下式计算产生:

$$P_\sigma = \alpha \times P_{\sigma-1} \quad \sigma = 1,2,\cdots,63 \tag{2}$$

$$\alpha = \left(\frac{0.01875}{0.5}\right)^{\frac{1}{63}} \text{且 } P_0 = 0.5 \tag{3}$$

并可由 P_σ 和 $\bar{\omega}$ 组成概率模型,$\bar{\omega}$ 为最大概率符号(MPS)的值,因其值取 0 或 1 中概率大者,故概率状态空间共有 128 个状态。这样预先计算好的 LPS 概率值 P_{LPS} 和 Q_ρ 就组成一张 64 × 4 大小的表 $TabRangLPS[\sigma,\rho]$,然后通过概率索引值 σ 和量化区间索引值 ρ 即可得到 R_{LPS}。编码器的流程如图 2 所示,其第 1 步先完成区间细分功能,然后根据 *binVal* 选择更新后的编码区间,流程图中的灰色部分负责完成概率更新,编码器最后是编码区间的归一化操作,其用于防止编码区间溢出。

CABAC 的计算开销主要集中在概率的估计、更

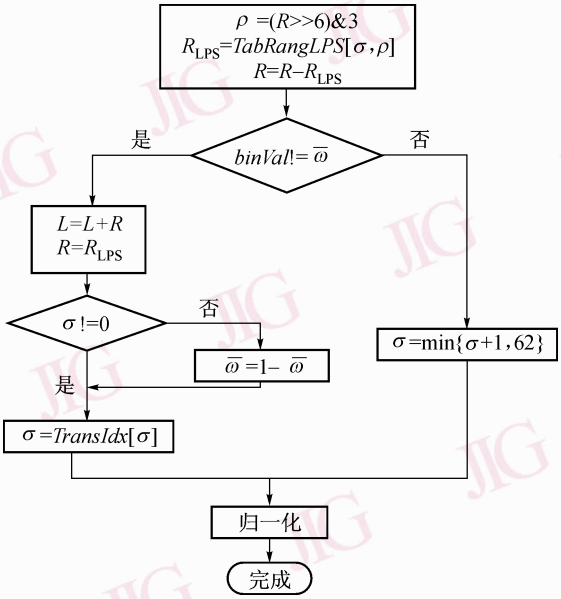

图 2　原始 CABAC 编码流程图

Fig. 2　Original CABAC coding flow

新以及细分编码区间时的运算两个方面。在该 CABAC 中,前者可通过有限状态机实现,而后者则通过移位、逻辑运算结合量化查表等方法实现。由于该编码器用这些手段代替了复杂的概率估计和乘法运算,因此在计算量和编码速度上有了较大改进,但是与 UVLC 相比,CABAC 的计算复杂度还是相当高的,在一些对编码速度要求较高的应用场合,CABAC 的应用仍受到很大的限制,可见进一步提高 CABAC 的编码速度有着很大的实际价值。

在 CABAC 中,每编码完一个符号后,都需要进行概率状态的更新,这样做的目的是为了使得编码器中上下文模型的概率估计能够更加符合视频流实际的概率统计特性,以提高编码效率。从图 2 可以看到,概率更新在整个编码流程中占了相当大的比重,由于每编码完一个符号都要进行概率状态的更新,因而就大幅度增加了 CABAC 的计算量。因此从提高 CABAC 编码速度的角度出发,概率估计的更新可以在每编码完一个包后进行,并使每个包中含有 N 个符号。这样虽然会牺牲一些概率估计的精度,但是却可以成倍减少概率估计的计算量。

3　改进的 CABAC 编码算法

图 3 是新算法的编码流程图,图中灰色部分负责完成概率更新,与经典的 CABAC 编码流程[2]相比,改进后的编码器在编码完一个包后才进行一次概率状态的更新,Packet_end 为包结束标志符。当 Packet_end 值为 1 时,则说明一个包的符号已编码完成,编码器就跳入概率估计更新分支;当 Packet_end 值为 0 时,则直接跳过概率估计更新这一步,进行归一化操作。

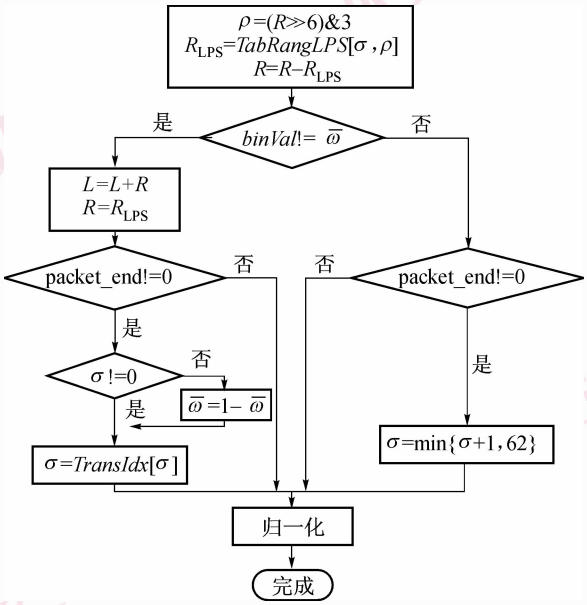

图 3　改进的 CABAC 编码流程图

Fig. 3　Modified CABAC coding flow

改进后的 CABAC 编码算法具体流程步骤如下(编码一个符号):

(1)根据当前的概率估计,对编码区间进行细分。如图 3 最上面的方框所示,这一步包含了以下 3 个操作:①根据编码区间 R 和式 $\rho = (R >> 6) \& 3$ 计算首先得到量化区间的索引值 ρ;②将上一步得到的 ρ,结合概率索引值 σ 表格 $TabRangLPS[\sigma,\rho]$ 来得到与 LPS 相关的编码区间 R_{LPS};③根据 $R = R - R_{LPS}$ 计算出输入符号 $binVal = \bar{\omega}$ 时的编码区间值;

(2)判断输入符号 $binVal$ 是不是 MPS,如果是,则编码器选择编码区间的左边子区间,此时编码区间 R 沿用第 1 步的计算结果,区间下限 L 不变(如图 3 右边分支所示);如果不是,则编码器选择编码区间的右边子区间,编码区间 $R = R_{LPS}$,区间下限 $L = L + R$(如图 3 左边分支所示);

(3)根据 Packet_end 的值来判断是否需要进行概率更新操作。如果 Packet_end 值为 1,则说明已编码完一个包的符号,可进入概率更新分支(如图 3 所示的灰色部分);如果 Packet_end 值为 0,则跳过

概率更新,进入第(5)步;

(4)进行概率估计的更新。对于 $binVal = \bar{\omega}$ 的情况,概率索引值更新为 $\sigma = \min\{\sigma + 1, 62\}$(如图 3 右边分支的灰色部分所示);对于 $binVal$ 为 LPS 值的情况,概率索引值可通过查表 $\sigma = TransIdx[\sigma]$ 得到更新,同时,根据 σ 是否为 0 来判断 LPS 值与 MPS 值是否需要互换,如果 σ 为 0,则说明 LPS 值的概率将大于 MPS 值,这时需要通过 $\bar{\omega} = 1 - \bar{\omega}$ 互换两者的值,否则不变;

(5)编码区间 R 和区间下限 L 的归一化操作,以防止区间溢出。

通过以上 5 步就完成了改进后的 CABAC 算法对一个符号的编码。

经典 CABAC 编码可以作为本文算法在包的大小 $N = 1$ bit 时的一个特例。

4 实验结果与分析

实验是基于 H.264 测试模型 JM9.5[6],其中帧率设为 30 fps,编码过程中没有选择率失真优化,配置文件中的 RDOptimization 设为 0,选择编码出来的序列为 IPPP…,每个测试序列编码 50 帧,所用的测试序列为 CIF 格式。实验中,I 帧和 P 帧的量化参数(QP)取值相同。实验所用计算机的配置为:CPU 为 Pentium(R) 4 2.8 GHz,512 MB 内存,操作系统为 Windows XP。包的大小 N 分别为 8 bits 和 16 bits,实验得到的改进算法与原始 CABAC 算法在编码速度(编码时间)和编码效率(编码出来序列的码长)上的性能如表 1 所示。

从表 1 可以看到,当包的大小为 8 bits, QP 值为 16 时,编码速度上升了 6.4% ~17.9%,平均为 13.3%,编码效率仅下降了 1.43% ~2.99%,平均为 1.87%;当包的大小为 8 bits,QP 值为 28 时,编码速度平均提高 18.8%,编码效率平均下降 2.18%;当包的大小为 16 bits,QP 值为 16 时,编码速度平均提高 24.9%,编码效率平均下降 2.44%;当包的大小为 16 bits, QP 值为 28 时,编码速度平均提高 30.7%,而编码效率则平均下降 2.98%。

从上面的实验数据分析可知,一方面,由于编码完一个包后,才进行概率更新,使概率估计与视频流的实时概率统计特性的符合性有所降低,从而导致了编码效率的下降,由于包大小 N 不是很大,因此编码效率的下降很小;另一方面,概率更新占了编码很大的时间,由于改进算法成倍地减少了概率更新所消耗的时间,从而有效地提升了编码速度,而且包大小 N 越大,编码速度提升得越多。

表 1 实验数据

Tab. 1 Experiment results

测试序列	QP 值	N(bits)	编码效率(%)	编码速度提高(%)
Akyio	16	8	-2.99	17.9
		16	-3.37	25.6
	28	8	-3.23	16.2
		16	-4.20	27.1
Carphone	16	8	-1.88	14.9
		16	-2.39	22.3
	28	8	-2.04	16.2
		16	-2.44	23.0
Coastguard	16	8	-1.43	10.2
		16	-2.03	19.9
	28	8	-1.76	13.5
		16	-2.55	28.9
Flower	16	8	-1.45	6.4
		16	-2.01	22.6
	28	8	-2.23	8.7
		16	-3.13	22.3
Football	16	8	-2.10	16.3
		16	-3.06	32.5
	28	8	-2.42	33.2
		16	-3.66	46.1
Forman	16	8	-1.78	17.4
		16	-2.38	26.8
	28	8	-2.11	21.2
		16	-2.90	33.8
Mobile	16	8	-1.44	10.3
		16	-1.86	24.5
	28	8	-1.47	22.5
		16	-1.98	33.7
平均值	16	8	-1.87	13.3
		16	-2.44	24.9
	28	8	-2.18	18.8
		16	-2.98	30.7

N 的大小直接关系到加速算法概率更新的滞后程度,即 N 越大,滞后越严重,概率估计的准确度就越差,其表现在编码性能上就是编码效率的进一步下降。

但是同样的 N,编码效率的下降也是不同的,这一点可以从表 1 清晰地看到。CABAC 算法是通过查表进行概率更新的,查表的依据是概率索引值 σ。

而 σ 的更新则是通过一个有限状态机进行的,其更新过程可用图 4 简单地表示,σ 状态转换的依据是当前编码的符号是 MPS 还是 LPS。当 σ 增大到 62 时,σ 保持不变;当 σ 减小到 0 时,MPS 与 LPS 的值进行交换。下面以 $N=8$ 为例来分析影响概率估计准确度下降的原因。本文选择 8 bits 长的两个序列{MPS,MPS,MPS,MPS,MPS,MPS,MPS,MPS,MPS}和{MPS,LPS,MPS,LPS,MPS,LPS,MPS,LPS}来进行概率估计。显然前面一个序列的概率估计准确度受到影响更大,因为概率索引值 σ 是一直增加的,此时后 7 bits 的概率估计都有偏差,而后一个序列则由于 MPS 和 LPS 交替出现,且 σ 的值在 σ 和 $\sigma+1$ 之间交替,因而只有 4 bits 的概率估计有偏差。因此当 N 相同时,MPS 的变化频度直接影响了概率估计的准确度,最好的情况是 MPS 与 LPS 交替出现,此时 MPS 变化最频繁,最差的情况是全为 MPS 或全为 LPS。同样的 N,而加速算法的编码效率下降不同的原因正在于此。

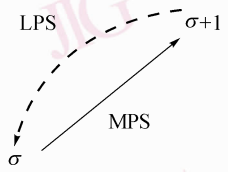

图 4　概率索引值 σ 的更新

Fig. 4　The update of probability index σ

实际应用中,可以根据需要选择一个合适的包大小,以兼顾编码效率和编码速度。另外,QP 值对结果也有一定的影响,QP 值越大,速度提升越可观,但是大的 QP 值也会引起编码效率的略微下降。

5　结　论

本文通过改进 CABAC 中概率估计的更新频率,提出了一种提高 CABAC 编码速度的有效方法,使 CABAC 的编码速度得到了较大的提升,但编码效率却下降较少。在移动多媒体通信等运算资源有限,但对编码实时性要求较高的场合,该改进算法显示出了良好的应用前景。

参考文献(References)

1　JVT-G050. ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC. Final Draft International Standard of Joint Video Specification[S].

2　Marpe D, Schwarz H, Wiegand T. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, **13**(7): 620-636.

3　Xie Lin, Yu Lu, Chou Pei-liang. Research on context-adaptive binary arithmetic coding [J]. Journal of Zhejiang University (Engineering Science), 2005, **39** (6):910-914. [谢林,虞露,仇佩亮. 基于上下文的自适应二进制算术编码研究[J]. 浙江大学学报(工学版),2005,**39**(6):910-914.]

4　Lin Jian-humg, Parhi K K. Parallelization of context-based adaptive binary arithmetic coders [J]. IEEE Transactions on Signal Processing, 2006, **54** (10): 3702-3711.

5　Ghandi M, Ghandi M M, Shamsollahi M B. A novel context modeling scheme for motion vectors context-based arithmetic coding [A]. In: Proceedings of Canadian Conference on Electrical and Computer Engineering [C], Niagara Falls, Canada, 2004: 2021-2024.

6　H.264/AVC Reference Software [CP/OL]. http://iphome.hhi.de/suehring/tml, ver JM 9.5, 2005.