# H.264/AVC 解码器优化的研究

刘鹏飞 刘 志 安 平 张兆杨
(上海大学通信与信息工程学院,上海 200072)

**摘 要** 视频编码新标准 H.264/AVC 与先前的标准如 MPEG2、H.263、MPEG4 第 2 部分相比,虽具有更高的编码效率和差错鲁棒性,但是这些编码效率的提高是以增加编码器和解码器的复杂度为代价的。研究表明,由于 H.264/AVC 编解码器的计算复杂度比其他视频压缩标准高出几倍甚至十几倍,因此实现实时编解码器需要寻找高效的优化方法。为了对解码器进行优化,从软件和硬件平台的角度提出了一种解码器的优化方法,最后进行了实验,实验结果显示,优化后的软件解码器能够达到实时解码。

**关键词** H.264/AVC 解码器 优化 数字信号处理
**中图法分类号**: TN919.81 **文献标识码**: A **文章编号**: 1006-8961(2006)11-1627-04

# Research of H.264/AVC Decoder Optimization

LIU Peng-fei, LIU Zhi, AN Ping, ZHANG Zhao-yang
(*School of Communication and Information Engineering, Shanghai University, Shanghai* 200072)

**Abstract** The new international video coding standard H.264/AVC has significant improvements in coding efficiency and error robustness in comparison with the previous standards such as MPEG2, H.263, MPEG4 part 2. But improved coding efficiency comes at the expense of adding complexity to coder/decoder. The research shows the conclusion that the calculation complexity of H.264/AVC codec has several times as the previous standards, so as to get real time codec we must find high-effective optimization methods. The paper introduces the optimization solutions on software and hardware. The experimental outcomes are shown to achieve real time decoding with the optimized decoder.

**Keywords** H.264/AVC, decoder, optimization, digital signal processing(DSP)

## 1 引 言

H.264/AVC 有实用性、移动性和 IP 网络的适应性 3 个方面的技术特点,并在混合编码的基本框架下,其关键模块做了重大改进,如多模式运动估计、帧内预测、多参考帧预测、基于内容的变长编码、2 维 4×4 整数变换、1/4Pixel 精度预测、去块滤波等[1]。总之,H.264 高效编码是以较高的运算复杂性和对硬件的高要求为代价的,如果编码器包含 H.264所有功能,那么编码的计算复杂度至少是 MPEG4 或 H.263 的 10 倍以上,而解码复杂度则大约相当于 H.263 的 2 倍。测试结果表明,H.264 Baseline 解码器将比 H.263 Baseline 解码器增加 2.5 倍以上的复杂度。因此,H.264 编解码的速度和性能优化关系到 H.264 的应用和推广。本文将从软件和硬件平台两个方面介绍解码器的优化方案,并对优化后的解码器进行了测试,测试结果表明,优化后解码器能够实时解码。

## 2 软件解码器的优化方案

基于软件的 H.264/AVC 实时视频编解码系统与硬件系统相比,不仅成本低,更重要的是软件的升

**基金项目**:国家自然科学基金重点项目(60332030);上海市科委重点攻关项目(055115008)
**收稿日期**:2006-07-05;**改回日期**:2006-08-03

**第一作者简介**:刘鹏飞(1982~ ),男。2004 年获得中原工学院电子信息工程专业学士学位,现在上海大学通信与信息工程学院攻读硕士学位。主要研究方向为数字视频处理。E-mail:lpf_815@163.com

级容易,灵活性强。JVT 根据每一个 H.264 工作草案的版本都推出一个编/解码器的参考软件— JM 系列,但是由于它主要用于测试目的,其代码运行效率比较低,远不能实现实时软件编解码的要求,因此实现实时编解码器需要采用高效的优化方法。H.264/AVC 软件解码器优化的方法一般包括算法优化、代码优化和使用 SIMD(single instruction multiple data)指令集进行优化。以下将分别阐述这 3 方面的优化方法。

### 2.1 算法优化

对于一个 H.264/AVC 解码器而言,主要完成以下几个任务:熵解码、帧内预测或运动估计(其中包括 1/4 Pixel 精度内插)、反量化、整数反变换和去块滤波。在多个测试序列上对各个模块进行测试的结果表明,去块滤波和非整数像素运动估计中的插值运算占用整个解码计算量的 33% 和 25%,码流解析和熵解码占 13%,整数反变换和重构占 13%[2]。由此可见,算法优化一般应集中在去块滤波、运动补偿、熵解码和整数反变换这几个模块上。

#### 2.1.1 去块滤波的优化

由于 H.264 中最小块的大小是 4×4,所以在每个 4×4 块中的每个点有相同的边界门限。依据这一点,16 个点可以只计算 4 次边界门限,其相应的滤波操作也可以 4 次完成。通过这种优化方法就可以将去块滤波函数的调用次数降下来。

#### 2.1.2 运动补偿的优化

(1)因为运动补偿时要先判断运动估计是整像素还是非整数像素,如果是非整数像素,则还要判断水平方向和竖直方向分别是 1/2 pixels 还是 1/4 pixels的插值,所以条件判断很多,但可根据避免嵌套判断语句的原则来简化条件判断语句;

(2)对内部像素可以做一些预先运算,以避免重复运算,对边界像素则单独处理;

(3)通过直接对整数像素进行矩阵运算来得到非整数像素插值结果,而利用 SIMD 技术则很容易快速处理矩阵运算。

#### 2.1.3 熵解码的优化

熵解码的优化主要是对 CAVLC(content-based adaptive variable length coding)码表的优化。由于 CAVLC 需要查找 7 张表,而每一张表又有很多空缺项。如果用 2 维数组实现查找表,则一方面浪费不少存储空间,另一方面由于无用项的存在,还会浪费一定的查找时间,因此所采用的优化策略是,将 2 维码表转换为二叉树码表,并根据码表内容进行分组,再采用高效的查找算法来提高码表的查找效率。

#### 2.1.4 整数反变换的优化

对整数反变换模块进行运算分析,分析结果如表 1 所示。由表 1 可见,由于算术运算的比例达到 80%,所以优化重点是减少算术操作。

**表 1 整数反变换模块运算次数**

**Tab.1 The number of IDCT calculation**

| 测试模块 | 总计 | 比例(%) |
|---|---|---|
| 算术 | 1.33e7 | 80 |
| 比较 | 1.62e6 | 10 |
| 移位 | 4.05e5 | 2 |
| 自加 | 1.21e6 | 7 |

**表 2 整数反变换优化前后时间对比**

**Tab.2 Results of IDCT Optimization**

| 宏块数目 | 优化前变换时间(ms) | 优化后变换时间(ms) |
|---|---|---|
| 396 | 431 | 20 |
| 1 584 | 1 930 | 110 |
| 6 336 | 8 122 | 411 |
| 8 192 | 12 929 | 500 |
| 19 200 | 21 320 | 1 082 |

一次反变换需要进行 128 次乘法和 96 次加法运算。当反变换进行 Hadamard 变换时,则需要进行 256 次乘法和 192 次加法运算。这样即使是 QCIF 图像,其逆变换也需要进行 50 688 ~ 101 376 次乘法和 38 016 ~ 70 632 次加法运算。由于乘除运算比加减法运算耗时多,所以整数反变换的优化应集中在减少乘法运算次数上。

通过观察变换矩阵可发现,该矩阵只包括 ±1 和 ±1/2 这 4 种系数,对于乘以 ±1 的系数来说,可将其转换为简单的加减法,而对于乘以 ±1/2,则可以先进行移位运算然后再进行加减法运算,这样就完全避免了代价昂贵的乘法和除法运算。另外可利用蝶形变换对矩阵相乘再做进一步改进。例如对 5 帧不同格式图像比较其优化前后的整数反变换过程,其所要的计算时间如表 2 所示。由表 2 可见,经统计优化后的反变换速度能提高 23 倍左右。

### 2.2 代码优化

代码优化主要包括程序结构优化、循环展开、数据类型选择等多种方法。

可通过调整函数结构来减少其中的判断和跳转语句,以加快解码速度。比如 JM 解码器中 decode_one_macroblock 函数,可以根据宏块类型采用不同的函数来解码宏块。还可以把结构简单但是调用频繁的函数改为内联函数,并嵌入汇编语言。

对于循环语句,主要从以下 3 个方面进行优化:

(1)减少循环本身操作耗时

由于每次循环都需要不停地对循环条件进行判断,因此当循环内部语句较少,操作耗时与循环操作本身相比,耗时比例过小时,则可以采取将循环展开或者部分展开的方法,以尽量避免这种情况的发生。

(2)减少数据相关性

当对同一个变量进行连续多次操作时,由于数据相关性,致使处理器无法把相邻的两条代码放到并行的两条流水线中执行,而只能等到通过前一个指令操作得到第 2 个指令所需的变量后,再继续进行下一步的计算,因此可以采取隔行执行的方法,以避免相同变量的多次引用。

(3)提高并行性和重用性

由于优化的目的是在一定的时间内,尽可能多地处理数据,因此优化后的循环操作需要保证并行处理能力,并能将计算得到的中间结果予以保留,以提高数据的利用率。

### 2.3　使用 MMX/SSE 指令集进行优化

在处理图像、声音等多媒体数据时,往往有大量的变换、矩阵计算等类型的运算,这些运算对于多个数据流常常执行相同的操作。针对这种运算的特点,Intel 推出了一些新的指令集,如 MMX(multimedia extension)和 SSE(stream SIMD extensions)等。它们都可以在单个指令的运行过程中,同时处理多个数据。MMX 指令是依靠 8 个 64bitsMMX 寄存器进行并行定点数计算,SSE 则可以在 8 个 128bitsXMM 寄存器中进行并行浮点运算[3]。

凡是涉及到矩阵运算的函数都可用 MMX/SSE 指令进行改写。根据 H.264/AVC 解码器中各个模块的特点,可用 MMX/SSE 指令优化的模块有:非整数象素的插值以及 4×4 整数反变换。

## 3　硬件平台的优化方案

由于目前的视频处理方案已经从 ASIC(application specific integrated circuit)方案转向 DSP(digital signal processing)平台,所以解码器硬件实现平台的优化主要是 DSP 平台上的优化。

在 PC 机上将编译执行的 H.264/AVC 解码器代码移植到 DSP 上,由于没有利用 DSP 的各种性能,致使运行效率很低,因此必须结合 DSP 本身的特性,对其进行优化才能实现实时解码。优化共分为项目级优化、C 语言级优化和汇编语言级优化 3 个层次。

### 3.1　项目级优化

对整个项目进行编译链接生成 DSP 代码时,进行合理的参数(*-mw　-pm　-o3　-mt* 等)选择。例如在项目编译时,可联合使用*-pm* 与*-o3* 编译选项来调用最高级别的软件流水线优化,这就增大了软件编译成 DSP 代码的并行性,并改善了循环代码的编排。

### 3.2　C 语言级优化

针对采用的 DSP 的具体特点进行数据结构的优化、循环的优化和代码的并行化处理。

(1)数据结构优化是根据 DSP 自身的特性进行的优化,其中包括数据结构的设计、cache 的设定、DMA(direct memory access)的运用等等。对于通用的 DSP 尤其需要在这方面进行优化,若安排常用数据至片内存储器,则可以大大降低访问时间,以提升运行的效率。

(2)重新定义数据类型。在避免运算过程中数据溢出的前提下,应尽可能使用短尺寸的数据类型,因为这既可以节省代码空间,又可提高运行速度。同时,尽可能用多位的指令来访问少位的数据。例如使用 int 型(32bits)指令访问 2 个 short(16bits)数据,即将其分别放在 32bits 寄存器的高 16bits 和低 16bits 字段,这种称为数据打包处理技术的访问方式可以提高一倍的数据读取效率,因其可减少对内存的访问次数,从而可减少运算耗时。

(3)通过循环分解来实现循环的优化。这是由于 DSP 的并行化运算程度很高,如果程序中判断跳转过多或者程序的循环嵌套的深度过多,则都将严重影响 DSP 发挥其并行化效果,因此应该用循环分解方法将多重循环分解为少循环或单循环,并以此提高指令的并行执行程度,进而提高代码的执行效率。

### 3.3　汇编语言级优化

由于 C 编译器只能完成 70% 的工作,因此通常这个阶段的 C 代码在 DSP 上运行效率仍然不高,为进一步改进性能,对运算量较大的模块需要进行汇

编语言级的优化,即使用汇编语言或者线性汇编语言来对某些运算量大的函数进行重新编写。

## 4 实验结果及分析

经过算法优化和代码优化后的解码器性能测试如表 3 所示。测试平台为 CPU P4 2.0G、内存 512MB。

**表 3 优化后解码器解码速度**

**Tab. 3 Results of optimized decoder**

| 序列 | 解码速度(fps) | |
|---|---|---|
| | $QP=30$ | $QP=32$ |
| Foreman_cif | 149.8 | 151.3 |
| Mobile_cif | 115.7 | 116.9 |
| Wall_D1 | 42.6 | 43.7 |
| Basketball_D1 | 26.9 | 27.1 |

由表 3 可知,优化后的解码器可以将 D1 序列进行实时解码。同一编码序列用 JM 8.5 和文中解码器分别解码后的 YUV 图像如图 1 所示,由图 1 可见,优化后解码器与参考解码器解码图像的质量相同。

(a) JM 解码的 Mobile 第 61 帧

(b)优化后解码的 Mobile 第 61 帧

(c)JM 解码的 Basketball 第 12 帧

(d)优化后解码的 Basketball 第 12 帧

图 1 解码后的 YUV 图像比较

Fig. 1 Comparison of decoder YUV

## 5 结 论

实验结果表明,优化后解码器不仅能够实现实时解码,并且优化后的解码器可实现与参考解码器完全相同的功能和效果,即非失真优化。

以后工作是进一步在硬件平台上进行优化,以达到实时解码。

### 参考文献(References)


1 Wiegand T, Sullivan G, Bjontegaard G. Overview of the H. 264/AVC video coding standard[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, **13**(7):560 ~ 576.

2 Horowitz M, Joch A, Kossentini F, *et al*. H. 264/AVC baseline profile decoder complexity analysis[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2003, **13**(7): 704 ~ 716.

3 Lee Juyup, Sung Moon, Sung Wonyong. H. 264 decoder optimization exploiting SIMD Instructions[A]. In: Proceedings of the 2004 IEEE Asia-Pacific Conference on Circuits and Systems[C], Tainan, Taiwan, China, 2004: 1149 ~ 1152.