# H. 264 实时编码的指令 Cache 优化

宋立锋, 戴青云
(广东工业大学信息工程学院, 广东广州 510006)

摘 要: 为了提高指令装载效率以达到实时编码, 本文提出两项 H. 264 编码的指令 Cache 优化措施: 一是调整编码过程以避免宏块编码循环体内的指令码数量大于指令 Cache 容量, 为此提出一种新的编码过程, 把通常编码过程的一个帧层宏块编码循环拆分成三个 Slice 层宏块编码循环, 同时运动估计与模式选择的过程与结果不变; 二是恰当地压缩代码, 即根据新编码过程三个 Slice 层宏块编码循环体内代码长度和指令 Cache 缺失的不同情况, 分别侧重于压缩代码与展开代码这两种截然相反的优化措施中的一种.

关键词: 视频编码; H. 264; 媒体处理器; Cache 优化
中图分类号: TN919. 81 文献标识码: A 文章编号: 0372-2112 (2008) 08-1615-05

# Instruction Cache Optimization on H. 264 Real-Time Encoding

SONG Li-feng, DAI Qing-yun
(*College of Information Engineering, Guangdong University of Technology, Guangzhou, Guangdong 510006, China*)

**Abstract:** In order to improve the efficiency of instruction loading into DSP, two techniques of instruction cache optimization on H. 264 encoding are proposed in this paper. As the first technique, adjust encoding procedure to prevent the quantity of the instruction codes inside marcoblock encoding loop over the capacity of instruction cache. Thus a new encoding procedure is proposed which fragments the conventional encoding procedure from one macroblock encoding loop at the frame level into three macroblock encoding loops at the slice level, while the results of motion estimation and encoding mode decision are unchanged. As the second technique, compact source code suitably. That is, emphasize adaptively either source code compacting or unrolling according to the length of source codes as well as the quantity of instruction cache miss.

**Key words:** video coding; H. 264; media processors; cache optimization

## 1 引言

最新视频编码标准 H. 264[1] 比之前的 H. 263 和 MPEG-4 提高压缩效率约 50%, 意味着 H. 264 仅需 H. 263 或 MPEG-4 一半的码率便可重建出同质量图像. 会议电视属于全双工实时多媒体通信, 在 CIF 分辨率下要求视频编码与解码的速度不低于 25 帧/秒, 方能提供流畅的图像播放和 150 毫秒内的传输延迟. H. 264 仍然沿用 H. 261 奠定的常规编码方法, 通过增加更多的编码模式来提高图像质量, 其编码运算量大约是 H. 263 的 4 ~ 5 倍[2]. 在嵌入式会议电视终端中用数字信号处理器 DSP 实现 H. 264 实时编码的难度非常大.

达成 H. 264 实时编码的途径是算法优化和实现方法优化. 本文讨论如何针对高速指令缓存器(Cache)优化编码程序以提高程序装载效率. 本文以用于 H. 323 会议电视终端的在 Philips 公司 TriMedia 媒体处理器上运行的 H. 264 编码库为实例, 提出两项指令 Cache 优化措施. 主要优化措施是调整编码过程以避免宏块编码循环体内的指令码数量大于指令 Cache 容量. 本文提出一种新编码过程, 把通常编码过程的一个帧层宏块编码循环拆分成三个 Slice 层宏块编码循环, 同时运动估计与模式选择的过程与结果不变, 从而显著减少指令 Cache 缺失次数, 同时保持压缩效率. 次要优化措施是恰当地压缩代码长度. 为了解决压缩代码长度的指令 Cache 优化措施与展开代码的处理器代码优化措施之间的矛盾, 本文建议在本文编码过程的三个 Slice 层编码循环体中, 根据代码长度和指令 Cache 缺失的不同情况, 分别侧重于这两种截然相反的优化措施中的一种, 以便尽快达成最佳状况.

以上措施大幅度减少指令 Cache 缺失引起的处理器停顿时间, 显著提高了 H. 264 编码速度. 在本文实例中, Cache 优化前指令执行时间即有效编码时间在总编码时间中仅占 30~ 40%, 处理器停顿等待从主内存装载指令和数据的时间超过 60%; 指令 Cache 优化后有效编码时间所占比重提高到 60~ 65%, 达到对 CIF 视频实时编码.

## 2 指令 Cache 优化措施

传统处理器(另类处理器是本世纪始浮现的流处理器[3])通过 Cache(一种介于 CPU 与主内存之间的高速

收稿日期: 2007-06-28; 修回日期: 2008-02-29

缓冲器) 来匹配相差很大的处理器主频与存储器存取速度. TriMedia 处理器[4] 内部为哈佛结构, 指令通道与数据通道分离. 其中 1300 系列的指令 Cache 容量 32K 字节, 1500 系列 64K 字节. 指令 Cache 块长 64 字节. 映射方式是 8 路组关联, 对于 1500 系列就是把 1024 个指令 Cache 块分成 128 个页面, 每页容纳 8 个块. 每次指令 Cache 缺失 CPU 停顿 29 个时钟周期. 指令 Cache 的更新法则仍是最近最少使用替代.

处理器读内存数据时, 如果 Cache 命中, 就直接从 Cache 装载数据入寄存器. 只有当 Cache 缺失时, 才启动内存读周期一次装载一段连续的数据( 称为 Cache 块) , 期间 CPU 处于停顿状况而不能执行后续指令, 直到被读取数据装入寄存器为止. 这段时间为 Cache 缺失停顿时间. CPU 恢复运行后, Cache 块的后续数据继续装入 Cache, 类似于独立于数据处理的后台操作.

指令 Cache 优化的目标就是减小指令 Cache 缺失次数及其引起的处理器停顿时间, 提高程序读取效率.

### 2.1 调整编码过程以避免循环体内指令码数量大于指令 Cache 容量

程序中循环体内的指令码执行多次, 最需要发挥 Cache 的缓存作用. 如果循环体内的指令完全装入 Cache, 指令 Cache 缺失和指令装入 Cache 的动作只发生在刚进入循环体的初次执行中, 以后指令留驻于 Cache 中, 不再发生 Cache 缺失和指令装入 Cache; 相反, 如果循环体内的指令码数量大于指令 Cache 容量, 在循环执行中不断遇到以前执行过而 Cache 保存不了的指令, 出现大量的与循环次数成正比的指令 Cache 缺失和 Cache 块更新, 严重降低处理效率.

指令 Cache 优化前采用图 1 的通常编码过程, 在一帧图像内逐个宏块编码并且生成宏块码流, 构成一个循环体, 循环次数为图像内宏块总数, 在结构上与码流一致, 便于码率控制和打包. 从目标文件长度了解到图 1 的宏块编码循环体的指令码数量远远大于 64K 字节. 于是指令 Cache 缺失停顿时间在编码总时间中所占比例达到 33% ( 平均值) , 与指令执行时间( 43.71% ) 相差无几, 参见图 3.

解决方法是拆分循环体, 包括: 与运动估计和模式选择有关的功能块置于第一个循环体, 其它功能块放在随后的循环体内, 使运动估计和模式选择的结果保持不变; 把宏块编码循环体所在层次从帧层改为 Slice 层, 使一帧图像分出多个 Slice. 得到图 2 的编码过程: 一个帧层循环体内包含从图 1 循环体拆出来的三个 Slice 层循

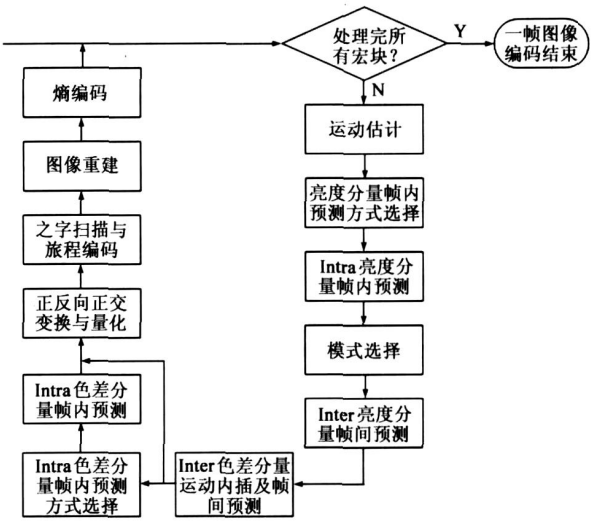


图 1 通常的编码流程 ( 一个帧层宏块编码循环 )

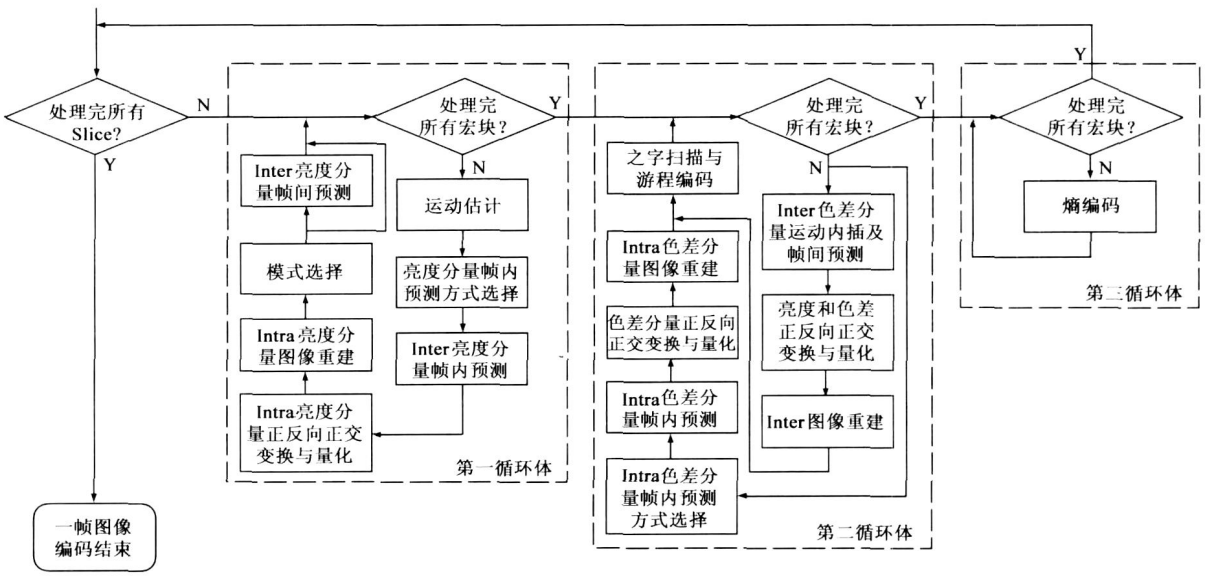


图 2 本文为指令 Cache 优化而提出的新编码流程 ( 一个帧层循环体内包含三个 Slice 层宏块循环体 )

环体; 在每帧图像编码前先划分出 Slice; 在帧层循环体内逐个 Slice 编码, 在 Slice 层循环体内逐个宏块编码, 循环次数分别是帧内 Slice 数和 Slice 内宏块数, 随每帧图像 Slice 划分的不同而变化. 该编码过程仅用于 P 帧编码.

原来集中于一个循环体的宏块编码功能块分散到三个循环体中. 其中第二、三循环体内的指令码数量已经小于 32K 字节, 第一循环体内的指令码数量也与 64K 字节差不多, 况且实际运行时并不需要对每个宏块执行全部指令. 算法优化后运动估计和模式选择过程包含了多种快速算法, 出现很多条件分支, 对大多数宏块仅执行第一循环体内的一部分指令. 因此三个循环体内的指令能以很高概率留驻于 Cache 中, 仅在第一循环体的处理中很少一部份宏块需要从主内存装载指令入 Cache.

图 2 编码过程的优点如下:

(1) 把一个指令过量的宏块编码循环体拆分成三个循环体, 每个循环体的指令码数量不超过指令 Cache 容量, 使得从主内存装载指令入 Cache 的动作只发生在刚进入循环体的初次执行中, 大幅度减少指令 Cache 缺失和从主内存装载指令入 Cache 的次数.

(2) 运动估计和模式选择的结果在宏块层达到最优, 保证了率失真性能.

缺点是: (1) 编码过程的结构与码流的结构不一致, 不便于打包和码率控制. (2) 不能在编码过程中根据码流长度划分 Slice, 只能在 P 帧编码前划分 Slice, 在不了解当前帧信源熵的情况下无法避免划分出的 Slice 的码流超长. (3) 亮度分量帧内预测时同一个 Slice 相邻 Inter 宏块的重建图像未知.

### 2.2　压缩代码

在硬件平台优化工作中, 压缩代码和展开代码是一对互相矛盾的措施. 后者包括展开、拆解直至消除循环体和循环体嵌套、重复写相同的代码段以消除条件判断分支及函数等; 前者则相反, 包括构造循环体和循环体嵌套、建立条件分支和函数以消除代码重复. 展开代码收益显著: 有利于提高 DSP 指令并行度和流水线执行效率, 减少跳转时现场保存与恢复的处理(如寄存器压栈和推栈). 但是代码展开会增加指令 Cache 的压力. 这部分工作属于低层次的代码优化工作. 停留在低层次的代码优化工作上不易达到两种措施之间的最佳平衡点.

根据指令 Cache 缺失在图 2 中三个循环体的分布侧重于不同优化措施. TriMedia 硬件 Profile 统计数据(TriMedia 处理器运行硬件或者软件仿真时汇报测量期间内所有函数或者决策树的时间开销) 显示, 用图 2 的编码过程处理 300 帧的 CIF 的 Foreman 序列生成 768kbps 码流, 在指令 Cache 缺失停顿时间中第一、二、三循环体所占的比例分别是 72.39%、4.02% 和 3.80%. 表明在第一循环体处指令 Cache 所面临的压力最大, 在第二、三循环体处则无关紧要. 于是在第一循环体内偏重于压缩代码, 减轻指令 Cache 所面临的压力; 在第二、三循环体内偏重于展开代码, 减少有效编码时间.

### 2.3　调整编码过程后需要解决的问题

#### 2.3.1　包长受限

IP 网最大允许传输单元 MTU 是 1.5K 字节. 本文实例要求 H.264 网络提取层单元 NALU 长度不得超过 1397 字节.

采用图 2 编码过程, 就不能根据累增的码流长度随时截断码流而划分出包含整数个宏块的 Slice, 只能在 P 帧编码前划分 Slice. 在不了解当前帧信源熵的情况下无法避免划分出的 Slice 的码流超长. 本文提出一种减少超长包出现次数的 Slice 划分方法, 把码率控制算法确定的每帧图像目标码长与一个常量(如 1200 字节) 比较, 得到 Slice 数, 该常量为一"不太长"的码流长度, 接着用前一个 P 帧所有宏块的预测均方差 MSE 预测当前帧信源熵, 以等熵值的原则划分 Slice, 力图减少 Slice 码流超长机会. 实际效果是: 场景中运动不大时, 帧间差异较小, Slice 码流基本上不会超长; 场景出现剧烈运动时, 超长包的出现概率依然很大, 需要用拆分包或二次 Slice 划分的方法限制包长.

应用 IETF RFC3984[5] 定义的拆分包把一个超长包拆分成多个不超长包, NALU 类型 28, 含 2 字节负载头和从原始 NALU 分割出的一段整数字节长度(未必包含整数个宏块) 的 Slice 码流. 在接收端根据拆分包的负载头和 RTP 字头内的序号把多个拆分包负载合并成为一个完整的 NALU, 再送去执行后续的解码与图像重建.

为了与不支持 RFC3984 的解码器互通, 以原来的 P 帧编码前的 Slice 划分过程为第一次划分, 在图 2 第三循环体中加入根据码流长度截断码流重新划分 Slice 的过程, 把第一次划分出的一个超长 Slice 第二次划分为多个不超长 Slice, 使得编码器发送的 RTP 包全是单 NALU 包. 对于在第二次划分中新划分出来的 Slice, Slice 边界的变化引起位于 Slice 边界的部份宏块的空间相邻性发生变化, 需要在熵编码时修改这些宏块的已经生成的编码结果. 对 Inter 宏块可能修改的编码结果是运动矢量预测值、宏块 Skip 运动矢量、Skip 宏块编码模式, 对 Intra 宏块需要重新编码. 在此增加的运算量相对很小, 损失的压缩性能也不大, 对总体性能无明显影响.

#### 2.3.2　帧内预测

图 2 中亮度分量帧内预测和 Intra 宏块的图像重建位于第一循环体, Inter 宏块的图像重建位于第二循环体. 亮度分量帧内预测时同 Slice 相邻 Inter 宏块的重建图像数据未知而同 Slice 相邻 Intra 宏块的重建图像数

据已知. 这种情况恰好与 H.264 规定的帧内预测受限的编码选项吻合, 由图像参数集中的 constrained_ intra_ pred_ flag 符号指示.

应用此编码选项实现图2的新编码过程, 附带增强了视频信源编码的容错性. 帧内预测受限, 率失真性能下降, 但是对总体性能影响不大, 因为 Intra 宏块在 P 帧出现概率甚低.

### 2.3.3 码率控制

图2中熵编码前 Slice 内所有编码模式、量化参数已定, 所有元素符号已经生成, Slice 的码流一次输出, 量化参数 QP 也只能一个 Slice 调整一次. 此外 2.3.1 的 Slice 划分方法划分出的 Slice 大小不断变化. 因此应用效果优良的 H.264 JM[6] 的码率控制算法时需要注意: 控制单元是大小不断变化着的 Slice; 量化参数 QP 只能一个 Slice 调整一次; 用 Slice 的绝对值和差 SAD/ Slice 面积计算均和差 MAD 时, Slice 面积是变量, 而非通常情况下是常量.

本文对码率控制算法所做修改是: 保存所有宏块的残差绝对值和 SAD; 用当前 Slice 在前一个 P 帧相同位置上所对应的所有宏块的 SAD 之和代入二次方率失真模型求量化参数 QP; 随后在更新二次方率失真模型时, 用当前 Slice 所有宏块的 SAD 之和以及当前 Slice 实际码长代入; 更新一次方均和差模型时, 则使用当前 Slice 所有宏块的 SAD 之和除以当前 Slice 内宏块数目.

修改后码率控制算法仍然保持 H.264 JM 算法的良好性能, 且运算量更低—只需要对每个 Slice 执行一遍码率控制的处理.

## 3 测试数据

测试平台为插在 PC 机 PCI 槽的 TriMedia 开发板, 使用4个标准测试序列的磁盘文件, Foreman, Mobile & Calendar, News, Silent Voice, 分辨率 CIF(352×288), 长度300帧. 从 main( ) 函数直入 H.264 编码模块, 单任务运行, 未使用 TriMedia 片内任何协处理器和外围部件, 从磁盘读入图像数据, 经 PCI 接口传输至 TriMedia 芯片, 编码后生成 H.264 码流输出到磁盘.

使用三个版本的 H.264 编码软件: Cache 优化前的版本, 指令 Cache 优化前的版本, 即 Cache 优化后的版本. 在 Cache 优化后的版本中, 通过拆分包使得输出视频包的长度不超过 1397 字节. 用 TriMedia 1300 开发板(处理器 PNX1302EH, 主频 200MHz, 普通 SDRAM 速度 100MHz)运行 H.264 编码的硬件仿真, 对300帧的测试序列执行编码, 额定码率 768kbps, 输出按各个函数分别统计的 Profile 文件, 读出其中的数据, 算出指令执行时间、数据 Cache 缺失停顿时间和指令 Cache 缺失停顿时间(三者之和即总编码时间), 列于图3.

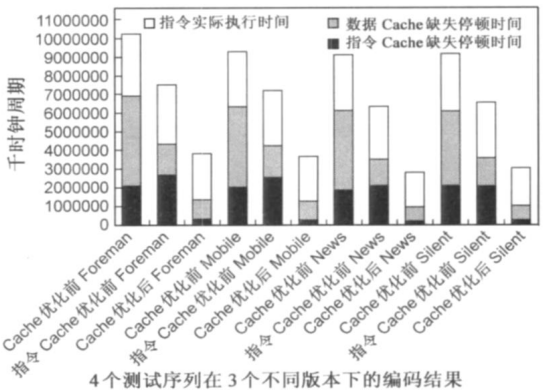


图3 TriMedia 1300 处理器完成 768kbps 的 H.264 编码的硬件 Profile 统计

再用 TriMedia 1500 开发板(处理器 PNX1502E, 主频 300MHz, DDR SDRAM 速度 400MHz) 对上面的指令 Cache 优化前的版本和最新版本测试编码时间(为不包含图像数据输入输出的单纯编码时间), 列于图4.

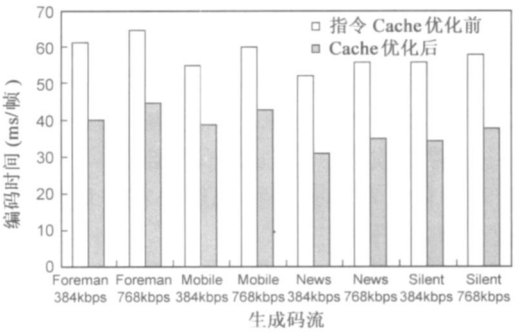


图4 TriMedia 1500 处理器完成 CIF H.264 编码的速度测试

最后测试指令 Cache 优化前的版本和最新版本的率失真性能, 列于表1, 反映编码过程调整前后(从图1变到图2)率失真性能的变化.

表1 编码过程调整前后的率失真性能比较

| 视频序列 | 采用图1通常编码过程 | | 采用图2新编码过程 | | 率失真性能差异 |
|---|---|---|---|---|---|
| | 码率(kbps) | Y PSNR(dB) | 码率(kbps) | Y PSNR(dB) | |
| Foreman | 386.73 | 35.26 | 382.51 | 34.91 | 码率增加4.22%或者质量下降0.17dB |
| | 515.71 | 36.46 | 510.77 | 36.19 | |
| | 772.13 | 38.13 | 761.72 | 37.98 | |
| | 1029.26 | 39.14 | 1017.46 | 38.98 | |
| Mobile & Calendar | 408.76 | 26.90 | 403.03 | 26.83 | 码率增加0.81%或者质量下降0.04dB |
| | 602.36 | 28.51 | 602.04 | 28.46 | |
| | 843.02 | 30.04 | 841.89 | 29.99 | |
| | 1201.62 | 31.73 | 1199.83 | 31.69 | |
| News | 386.61 | 41.32 | 381.33 | 41.12 | 码率增加1.28%或者质量下降0.06dB |
| | 515.00 | 42.76 | 509.99 | 42.60 | |
| | 769.81 | 44.56 | 747.37 | 44.41 | |
| | 1024.14 | 45.59 | 1017.24 | 45.66 | |
| Silent Voice | 385.80 | 38.68 | 380.72 | 38.21 | 码率增加4.98%或者质量下降0.26dB |
| | 515.31 | 40.16 | 508.17 | 39.78 | |
| | 771.90 | 42.23 | 761.86 | 41.93 | |
| | 1029.65 | 43.58 | 1016.08 | 43.50 | |

## 4 结束语

指令 Cache 优化后, 在 TriMedia 1300 芯片上, 指令 Cache 缺失停顿时间少了一个数量级, 在总编码时间已经缩短至原来的 47.92% 的情况下, 指令 Cache 缺失停顿时间的比例还是从 33.03% 降低到 7.48%; 在 TriMedia 1500 芯片上, 总编码时间已经缩短至原来的 65.92%. 新编码过程率失真性能下降幅度很微小, 平均 0.13dB, 最大 0.26dB. 证明了本文方法的有效性及实用价值. 指令 Cache 优化后的 H.264 编码帧频, 在 TriMedia 1300 芯片上, 384kbps 码率下从 16 帧/秒提高到 24 帧/秒; 在 TriMedia 1500 芯片上达到在 2Mbps 码率下全线实时. 包含本文指令 Cache 优化成果的 H.323 会议电视终端于 2005 年中在 H.264 编码的图像质量、帧频等指标上均达到当时的业界先进水平.

参考文献:

[1] ITU-T, H.264, Advanced video coding for generic audiovisual services[S]. March 2005.

[2] 薛金柱, 沈兰荪. 一种基于 H.264/AVC 的高效块匹配搜索算法[J]. 电子学报, 2004, 32(4): 583-586.
Xue Jin-zhu, Shen Lan-sun. An efficient block-matching motion estimation algorithm for H.264/AVC [J]. Acta Electronica Sinica, 2004, 32(4): 583-586. (in Chinese)

[3] Kapasi U, et al. The imagine stream processor[A]. Proceedings of the 2002 IEEE International Conference on Computer Design [C]. Freiburg, Germany: IEEE Computer Society, 2002. 282-288.

[4] Philips Electronics North America Corporation. TM1300 Preliminary Data Book[R]. North America: Philips, 1999.

[5] Internet Engineering Task Force. Request for comments 3984: RTP payload format for H.264 video[S]. February 2005.

[6] K Lim, et al. Text description of joint model reference encoding methods and decoding concealment methods, JVT of ITU-T VCEG and ISO/IEC MPEG Document JVT-K049[Z/OL]. http://ftp3.itu.int/av-arch/jvt-site/2004_03_Munich/JVT-K049.doc, 2005-01-17/2005-08-22.

作者简介:

宋立锋　男, 1967 年生于陕西西安. 于 1989 年、1992 年和 2003 年获得华南理工大学学士、硕士和博士学位. 2003.4~2007.6 作为东南大学无线电工程系的博士后在中兴通讯股份有限公司从事 H.264 产品开发以及可分级编码 SVC 技术研究工作. 现在广东工业大学信息工程学院任教. 研究兴趣为图像和视频处理及编码、多媒体通信等. Email: song.lifeng@126.com

戴青云　女, 1965 年生于湖南常德, 于 1991 年、2001 年获得华中理工大学硕士、华南理工大学博士学位. 现为广东工业大学信息工程学院教授. 研究方向为图像处理与模式识别、射频识别.