一种高性能低复杂度的基于串匹配的屏幕图像无损压缩算法

林 涛 蔡文婷* 陈先义 周开伦 王淑慧
(同济大学超大规模集成电路研究所 上海 200092)

摘 要：传统无损压缩算法对屏幕图像的压缩效果不佳。该文根据典型屏幕图像的特性，以 LZ4HC(LZ4 High Compression)算法为具体实现基础，提出一种基于串匹配的高性能低复杂度(String Matching with High Performance and Low Complexity, SMHPLC) 的屏幕图像无损压缩算法。相对于传统字典编码无损压缩算法，新算法提出了以像素为搜索和匹配单位，对未匹配串长度、匹配串长度以及匹配偏移量这 3 个编码参数进行联合优化编码，并对参数进行映射编码。实验结果表明，SMHPLC 具有高性能和低复杂度的综合优势，大幅降低编码复杂度，提高了编码效率。使用移动的文字和图形类的 AVS2 通用测试序列作为测试对象，对于 YUV 和 RGB 两种格式，SMHPLC 算法比 LZ4HC 总体节省码率分别为 22.4%，21.2%，同时编码复杂度降低分别为 35.5%，46.8%。

关键词：无损压缩算法；屏幕图像编码；字典编码；LZ4 High Compression (LZ4HC)
中图分类号：TN919.8 **文献标识码**：A **文章编号**：1009-5896(2017)02-0351-09
DOI: 10.11999/JEIT160560

Lossless Compression Algorithm Based on String Matching with High Performance and Low Complexity for Screen Content Coding

LIN Tao CAI Wenting CHEN Xianyi ZHOU Kailun WANG Shuhui
(*Institute of VLSI*, *Tongji University*, *Shanghai* 200092, *China*)

Abstract: Traditional lossless compression algorithms are not efficient for screen content coding. To take the full advantage of special characteristics of screen content, a lossless compression algorithm based on String Matching with High Performance and Low Complexity (SMHPLC) is proposed. It is implemented on the basis of LZ4HC (LZ4 High Compression). The main new ideas are using pixel instead of byte as the basic unit for string searching and matching, adopting joint optimal coding of three parameters of literal length, match length and offset and mapping for three parameters. Experiment results show that SMHPLC has both high coding efficiency and low complexity. Compared to LZ4HC, SMHPLC not only has a coding complexity reduction of 34.6%, 46.8%, but also achieve overall bit-rate saving of 22.4%, 21.2% in YUV, RGB color formats respectively for AVS2 common test sequences in moving text and graphics category.

Key words: Lossless compression algorithm; Screen content coding; Dictionary coding; LZ4 High Compression (LZ4HC)

1 引言

随着移动互联网的飞速发展以及虚拟化技术的日益成熟，在市场需求的推动下，移动平台和云计算平台[1]逐步得到广泛的应用。云计算平台作为传统计算机和网络技术发展融合的产物，通过整合分布式资源，可以把海量的数据存储和程序执行迁移到云端，提高安全性，降低成本和能耗[2]。作为云计算平台的典型应用之一，桌面云可以实现用户在瘦客户端或者其他与网络连接的设备(如普通台式机、笔记本电脑、智能手机等)通过网络访问数据中心云端服务器和应用程序[3]。用户由传统的终端至应用的访问模型变为从本地客户端连接到桌面云获取到应用并显示在屏幕上。因此，提高屏幕图像的传输效率是亟需解决的重要问题[4]。

屏幕图像包含自然图片和文本图形等。对于拍摄的自然图像，现有的图像和视频编码标准(如 JPEG, H.264/AVC, AVS, HEVC 等)具有出色的编码性能。但对于计算机产生的文本、图形、图标等区域，传统视频编解码器的效果并不理想。为了提高屏幕图像编码的性能，国际电信联盟 (ITU-T)、

收稿日期：2016-05-28；改回日期：2016-11-19；网络出版：2016-12-29
*通信作者：蔡文婷 caiwenting@tongji.edu.cn
基金项目：国家自然科学基金(61601200, 61271096)，高等学校博士学科点专项科研基金(20130072110054)
Foundation Items: The National Natural Science Foundation of China (61601200, 61271096), Specialized Research Fund for the Doctoral Program of Higher Education (20130072110054)

国际标准化组织(ISO)和国际电工委员会(IEC)联合启动了基于屏幕图像编码(Screen Content Coding, SCC)标准的研究。

对于屏幕图像中非连续色调内容，基于串匹配的字典编码有很好的压缩性能[5−8]。字典编码的基本原理是建立一个已经完成编码的历史数据的空间(字典)，在其中寻找待编码数据的匹配串(即参考串)，以编码匹配参数(匹配的位置，匹配的长度)替代当前数据串写入码流。当前主流的字典编码算法中，Gzip 和 zlib 有很高的压缩率但处理器资源消耗较大，速度较慢[9]。Bzip2 和 7zip 算法有很好的压缩效果，但对计算资源的消耗比 Gzip 和 zlib 还要高得多。LZ4[10−12]作为当今高性能的无损压缩算法的典型代表，编码速度远超 zlib 并且解码速度近乎是其 5 倍。它的编码速度能达到超过 400 MB/s，解码速度也能超过 1.8 GB/s。作为 LZ4 的高压缩率(High Compression, HC)版本[13]，LZ4HC 以更全面的搜索方式弥补了 LZ4 进行快速搜索而引起压缩比的损失，虽然编码时间有所增加，但解码时间不受影响，是目前商用屏幕图像编码产品的首选算法。

屏幕图像由像素构成。而传统的 LZ4 和 LZ4HC 等算法都是以字节为单位，将像素的 3 个分量看成 3 个独立的字节，未充分利用像素的冗余，尤其是匹配串的位置可能不是整像素的边界而降低了编码效率。

针对传统字典编码无损压缩算法的这些缺陷，本文提出了基于像素为单位的字典编码无损压缩算法，结合典型屏幕图像的特有统计特性，采用对匹配参数进行映射和可变字节联合优化编码的方式进一步提高压缩性能，同时也大大降低计算复杂度。

2 本文提出的方法

本文以 LZ4HC 算法为具体实现的基础，提出一种充分利用屏幕图像的数据特性的基于串匹配(String Matching)的高性能低复杂度(High Performance Low Complexity)屏幕图像无损压缩方法：SMHPLC 算法。

2.1 传统的 LZ4HC 算法的基本原理

LZ4HC 的压缩后码流的数据[14]由 4 个部分组成：未匹配串长度、未匹配串字节、匹配串长度及匹配偏移量。如图 1 所示，写入码流的数据依次为 1 个字节的 Token, 0 至多个字节的 Literal Length (未匹配串长度)，0 至多个字节的 Literals(未匹配串字节)，两个字节的匹配串 Offset(匹配偏移量)以及 0 至多字节的 Match Length(匹配串长度)。Token 的高 4 位表示 Literal Length，低 4 位表示 Match Length，取值范围均为 0~15。当未匹配串长度或者匹配串长度小于 15 时，则不需要消耗额外的字节，否则剩下的长度 1 次消耗 1 Byte 写入码流，直到写入的最后一个字节小于 255，每个字节表示的长度范围是 0~255。Offset 是当前串距离参考串的偏移量，以 Byte 为单位，范围是 1~65535。匹配串的默认最小允许长度为 4 Byte，因此将 Match Length 减去 4 后再进行编码，实际最小值为 0。下文中提到的 Match Length 均为实际编码的 Match Length。

LZ4HC 利用散列表(Hash Table)查找最佳匹配串，通过键值对(key, value)的映射关系进行搜索。默认的 Hash 表大小是 16 kB。Hash 表中存放具有相同 Hash 值且离当前待编码位置最近的参考点 P_n 与参考基(Base)之间的距离，如式(1)所示。LZ4HC 中还分配了链表(Chain Table)数组用来多次尝试匹配以获取最长匹配串，当前位置对应的参考索引(index)取低 16 位作为链表的下标，如式(2)，式(3)所示，计算 index 与 Hash 表中对应值之间的差值 delta，并将其存入链表数组中。下面提到的索引均为当前地址到 Base 的距离，默认的 Base 位置为输入的压缩数据块的起始位置减去 64 kB。默认链表使用内存是 64 kB，故 Offset 只需要两个字节。对当前待匹配串前面的字符串按每 4 Byte 采用 2^1 至 2^{32} 之间的黄金分割素数 2654435761U 计算 Hash 值 h。搜索 Level 可设定的最大值为 16，即最大搜索次数为 2^{15}。

$$\text{HashTable}[h] = P_n - \text{Base} \tag{1}$$

$$\text{delta=index-HashTable}[h] \tag{2}$$

$$\text{ChainTable[index\&FFFF]=delta} \tag{3}$$

搜索过程如下：

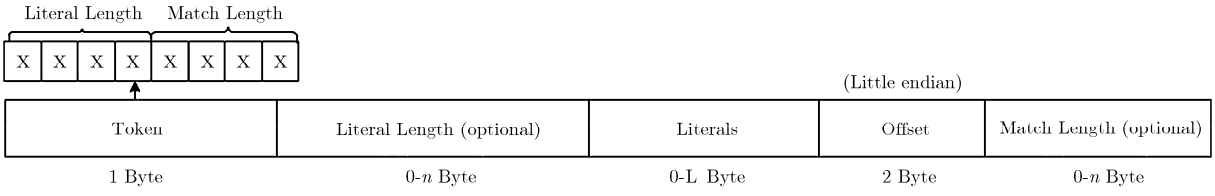

图 1 数据块的输出码流的格式

(1)当前地址指针 ip，当前地址索引为 index，上一次已编码的串的起始地址索引为 nextToUpdate。若 index<nextToUpdate，则以字节为单位增加 index，依次计算对应位置的 h，并将 delta 存入链表中。

(2)计算当前待编码位置的 Hash 值 hcur。最佳匹配长度 ml 置为 0。

(3)若该位置为首次寻找匹配，则根据 hcur 获取 Hash 表中的匹配位置；否则，从 Chain Table 中获取匹配位置。匹配位置索引 matchIdx，对应的地址为 ref。

(4)如果 ref 在限制条件的范围内且不超过搜索次数，

(a)如果 ref 等于 ip，继续计算匹配长度。获得该匹配位置的匹配长度为 mlt。若 mlt>ml，则 ml=mlt。

(b)否则跳转到步骤(3)。

不符合匹配前提条件，则跳转到步骤(5)

(5)按照图 1 数据块的输出格式编码 Literal Length, Literals, Offset, Match Length，写入码流中。

默认的最小匹配串长度是 4 Byte，若当前位置未搜索成功，则将当前指针 ip 右移 4 Byte 继续进行下轮搜索。默认设置下，当待编码串长度大于 13 Byte 的时候才会进行 Hash 搜索匹配，且最后 5 Byte 总是 Literals。传统的 LZ4HC 在 Hash 搜索后还会扩大范围进行第 2 次搜索，本文改进该算法时并没有采用。

2.2 SMHPLC 算法

针对屏幕图像特性和传统字典编码算法的缺陷，本文提出如下改进以提高编码效率和降低计算复杂度：

(1)将基于字节的搜索和匹配改为基于像素的搜索和匹配。

(2)对 Literal Length 和 Match Length 等匹配参数进行联合优化编码。

(3)建立映射数组，将匹配参数中出现频度很高且数值很大的区段映射为数值较小的区段。

SMHPLC 算法的整体框架如图 2 所示，主要分为更新 Hash Table 和 Chain Table，在最大搜索次数的限制内寻找最佳匹配，和编码 (包括 3 个编码参数，以及复制 Literals)这 3 个部分。

2.2.1 基于像素的搜索和匹配　对输入的编码块(也可以是整幅图像)按照水平或者垂直扫描顺序把图像数据排列成像素串，如图 3 所示，以 YUV 色彩格式为例，将 Y, U, V 3 个分量以该顺序打包成一

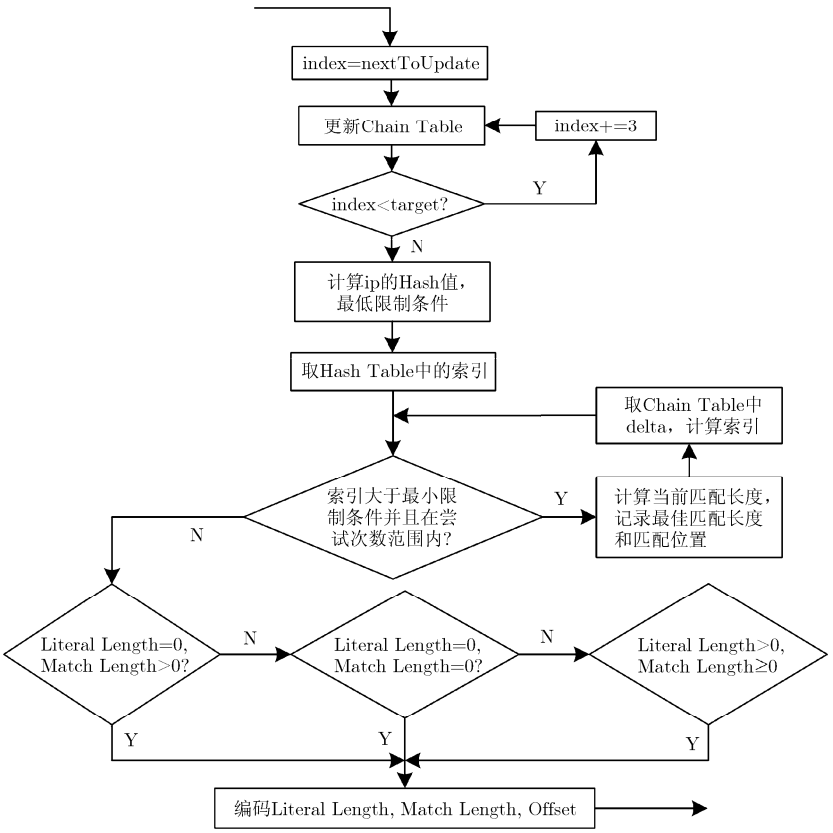

图 2 SMHPLC 算法的整体框架

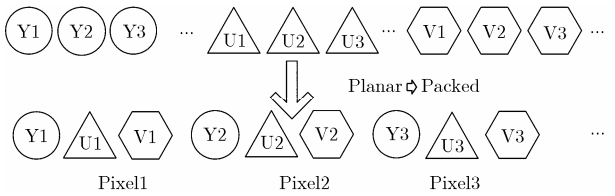

图 3 像素打包排列

个像素 Pixel(YUV)。基于此，本文将传统算法中连续 4 Byte 计算 Hash 值的方式改为计算 1 个像素的 Hash 值，即连续的 3 Byte 的 Hash 值。所以在进行 Hash 搜索前，以像素为单位增加 index，将位置信息存入 Hash Table 和 Chain Table 中。在搜索最佳匹配串时，匹配串的长度也以像素为单位来计算，最小匹配串长度则是一个像素(即 3 Byte)。这样，既能找到更长匹配串，也能将有关匹配参数的数值缩小为原来的 1/3，从而进一步提高编码的性能。

本文提出的方法对 YUV 色彩格式或者 RGB 色彩格式的屏幕图像均有效，同时对按照水平或者垂直扫描顺序排列的屏幕图像也都同样有效。对于大多数屏幕内容，水平扫描比垂直扫描的编码压缩性能相对高一些。但对于有些屏幕内容，如垂直线条比较多的屏幕图像，垂直扫描的编码压缩性能高一些。

2.2.2 对匹配参数的联合优化编码 使用 LZ4HC 和 SMHPLC，以 8 个移动的文字和图形类的 AVS2 屏幕与混合内容视频编码通用测试序列[15]为统计样本空间，对测试数据分别进行搜索 Level 为 4 时基于像素和字节匹配的 Literal Length 和 Match Length 的联合分布统计。联合分布结果如表 1 所示，按像素匹配时，Literal Length = 0 同时 Match Length = 0 的情形比例达到 24.10%，而按字节匹配时只占 4.12%。由此可见，按像素匹配时，如果沿用 LZ4HC 的方法，Token 的高 4 位和低 4 位均为 0，另外还消耗 2 Byte 编码 Offset，总共消耗 3 Byte 来编码这种匹配串。实际上，可能更好的策略是在 Token 中拿出 1 bit 或者 2 bit 作为 Flag，将剩下的比特用来编码 Offset，在某些情况下可以不需要额外的字节就能完成一个匹配串的编码。因此，本文进行了如下两种方案的尝试：

方案 1：

A 类 Token 中的最高位 0 表示 Literal Length 和 Match Length 均为 0 的情形；

B 类 Token 中的高两位 10 表示 Literal Length 等于 0 且 Match Length 大于 0 的情形；

C 类 Token 中的高两位 11 表示 Literal Length 大于 0 且 Match Length 不小于 0 的情形。

表 1 Literal Length 和 Match Length 联合分布(%)

SMHPLC (Pixel)		
P(Literal Length, Match Length)	Match Length=0	Match Length>0
Literal Length = 0	24.10	65.29
Literal Length > 0	5.32	5.28
LZ4HC (Byte)		
P(Literal Length, Match Length)	Match Length=0	Match Length>0
Literal Length = 0	4.12	79.96
Literal Length > 0	3.74	12.18

方案 2：

A 类 Token 中的最高位 0 表示 Literal Length 等于 0 且 Match Length 大于 0 的情形；

B 类 Token 中的高两位 10 表示 Literal Length 和 Match Length 均为 0 的情形；

C 类 Token 中的高两位 11 表示 Literal Length 大于 0 且 Match Length 不小于 0 的情形。

经实验，方案 2 优于方案 1，故本文采用了方案 2，并在此基础上继续对 3 个匹配参数的编码进行改进。

传统的 LZ4HC 编码 Match Length 和 Literal Length 都是用 1 Byte 表示 0～255，多个字节逐个累加的方式编码 1 个长度。对上述测试序列进行 Match Length 的统计，长度小于 256 所占比例高达 99%，除 Token 编码的长度外最多只需要 1 Byte 就可以编码 Match Length 的整个长度。因此，本文提出了根据掩码分段编码的算法(记为 MaskLevelExtend)，编码部分描述如表 2 所示。

这样，根据输入的编码参数值所在的掩码区间分成 4 段进行编码，以只利用 Token 中分配的比特数，或者还需要额外的 1 Byte、2 Byte 或更多字节进行编码，可以大幅减少字节消耗，提高数据压缩率。

基于像素匹配的 Offset 的最大值为 65535/3= 21845，经统计 Offset 小于 255 的部分占总数的 70.3%，因此为所有的 Offset 都分配两个字节造成了很大的浪费。针对 C 类 Literal Length>0 且 Match Length=0 和 Literal Length>0 且 Match Length>0 的情形，本文提出在 Token 中再利用 1 bit 作为 Offset 是否小于 255 的标志，当 Offset 小于 255 时只分配 1 Byte，以此进一步提高压缩率。对于 A 类和 B 类，由于 Literal Length 等于 0，可以将 Token 中除去标志位以外剩下的比特共同编码 Match Length 和 Offset，通过尝试不同的分配策略找出最佳方案。经试验，A, B, C 类中最终的 Token 比特分配如图 4 所示。

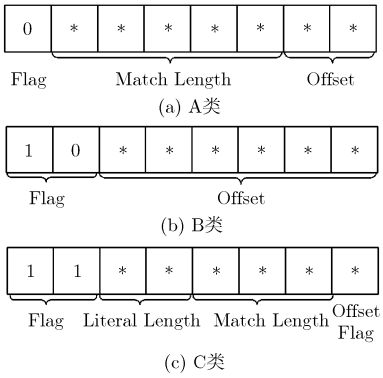

图 4 A, B, C 类中 Token 的比特分配

表 2 掩码分段编码的算法的编码

输入：VALUE_BITS　// Token 中 value 所占的比特位数
　　value　　//待编码参数值
输出：码流数据
　VALUE_MASK=(1<<VALUE_BITS)-1;
IF(value<VALUE_MASK-1)
　　在 Token 中指定的 VALUE_BITS 位置写入 value;
ELSE IF(value ≤ VALUE_MASK-1+255)
　　在 Token 中指定的 VALUE_BITS 位置写入(VALUE_MASK-1)，将 value-(VALUE_MASK-1)用一个字节的 char 类型表示写入码流;
ELSE
　　在 Token 中指定的 VALUE_BITS 位置写入 VALUE_MASK;
　　IF(value-VALUE_MASK<65535)
　　将(value-VALUE_MASK)用两个字节的 U16 (unsigned short)类型表示写入码流;
　　ELSE
　　将 65535 以 U16 类型写入码流，再将(value-VALUE_MASK-65535)以 char 类型逐个写入码流中;
　　ENDIF
ENDIF

2.2.3 对匹配参数的映射　映射的思想是将匹配参数中部分出现频度较高且需要较多字节编码的值映射为频度较小且字节消耗较少的值，从而减少总体字节消耗。根据对 Match Length 和 Offset 的频度统计，建立映射数组，从数组中取出该匹配参数值对应的映射值后，再对该映射值进行后续的编码。

本文采用了整体映射与局部映射相结合的方案，取值如表 3 所示。对于整体映射，考虑 Token 中各编码参数的比特分配以及频度统计，建立上文中 A, B, C 3 类情况均适用的 Offset 映射数组。对于局部映射，统计情况 A 中 Match Length 和 Offset 的联合频度，对满足映射要求的 Match Length 和 Offset 建立联合映射对，在编码参数前进行局部一一映射。由于情况 B 实际只编码 Offset，比较不同的掩码分段区间内出现的频度，选取合适的数值进行映射。映射的具体过程为，在进行分类编码前，首先对 Offset 进行整体映射，从映射数组中取得对应的映射值作为接下来编码的 Offset。然后在 A, B 两种情况下，对待编码的 Match Length, Offset 进行该情况下的局部映射，再对参数的映射值进行编码。由于情况 C 下 Token 中用于编码的 Match Length 的比特数较少，所以该情况未进行局部映射。

表 3 整体映射和局部映射取值

映射参数	整体映射(情况 A, B, C 均适用)	局部映射	
		情况 A	情况 B
Offset	(64，2), (1280，249) (1920, 215), (3840, 239)	(4040, 253)	(65, 49)
Match Length	无	(55，28)	无

在 LZ4HC 算法基础上实现的 SMHPLC 算法的流程如图 5 所示，Flag 的取值范围为 0, 2, 3，分别对应 A, B, C 3 类情形。

3　实验结果

本文的实验在 lz4-r127[16] 代码基础上实现 SMHPLC 算法，并且使用表 4 所示 8 个 AVS2 屏幕与混合内容视频编码通用测试序列的前 10 帧来测试和评估 SMHPLC 算法的性能和复杂度。

表 4　8 个 AVS2 屏幕与混合内容视频编码通用测试序列

序号	序列名	短名	分辨率	帧率	编码帧数
1	sc_flyingGraphics_1920x1080_60_8bit	FLYG	1920×1080	60	300
2	ClearTypeSpreadsheet_1920x1080_30_8bit	SPS	1920×1080	30	300
3	BitstreamAnalyzer_1920x1080_30_8bit	BSA	1920×1080	30	300
4	EnglishDocumentEditing_1920x1080_30_8bit	EDE	1920×1080	30	300
5	ChineseDocumentEditing_1920x1080_30_8bit	CDE	1920×1080	30	300
6	sc_console_1920x1080_60_8bit	CNS	1920×1080	60	600
7	sc_desktop_1920x1080_60_8bit	DSK	1920×1080	60	600
8	CircuitLayoutPresentation_1920x1080_30_8bit	CLP	1920×1080	30	300

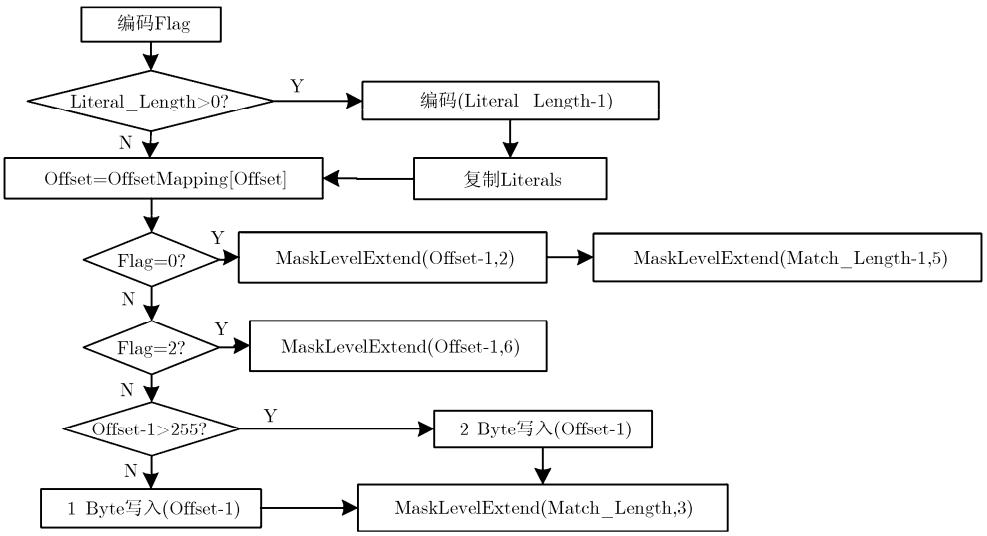

图 5 SMHPLC 算法编码流程

每个序列有 RGB 和 YUV 两种色彩格式。实验的硬件环境为 Intel(R) Xeon(R) CPU X5460 @3.16 GHZ。

实验比较了传统的 LZ4HC 算法，在其基础上实现的 SMHPLC 算法，zlib 算法和 LZMA 算法之间的性能和复杂度(使用 HEVC 标准制定工作组和 AVS 标准制定工作组规定的编解码运行时间来衡量复杂度)。

表 5 和表 6 是两种色彩格式下 SMHPLC 算法和传统的 LZ4HC 算法在相同的搜索 Level 下总体的压缩后码流比特率、编码时间和解码时间的比较。比较的搜索 Level 的范围为 4 至 14，随着搜索 Level 的增加，YUV 格式下 SMHPLC 算法与 LZ4HC 算法的比特率之比和解码时间之比均逐步减少，虽然编码时间的比值有一定的增大，但 SMHPLC 算法的编码速度依然远超过 LZ4HC 算法。RGB 格式下这两种算法的比特率之比、解码时间之比、编码时间之比与 YUV 格式下有相似的变化趋势。LZ4HC 本身是一种极低复杂度的算法，YUV 格式下 SMHPLC 算法减少了 35.5%～47.5%的编码时间，性能却提高了 17.3%~22.4%，RGB 格式下减少了 45.3%~51.3% 的编码时间，性能提升了 18.4%~21.2%。同时，SMHPLC 算法仍然保持着很快的解码速度，YUV 格式下解码时间只增加了 15.7%~29.9%，RGB 格式下解码时间只增加了 14.9%~25.7%。

表 5 YUV 格式下 SMHPLC 算法与 LZ4HC 算法的比较

Level	平均比特率(kbps)		SMHPLC 的比特率或时间/LZ4HC 的比特率或时间 (%)		
	SMHPLC	LZ4HC	比特率的比	编码时间的比	解码时间的比
4	105944.2	128088.0	82.7	54.1	129.9
5	98381.9	120997.8	81.3	53.0	126.7
6	92864.3	115634.4	80.3	52.5	123.4
7	88491.3	111573.7	79.3	52.8	119.5
8	86012.4	108527.9	79.3	54.6	119.3
9	84013.1	106777.1	78.7	55.3	118.8
10	82430.4	105484.9	78.1	56.8	118.5
11	81331.9	104318.0	78.0	60.0	117.8
12	80368.0	103365.2	77.8	63.4	116.2
13	79943.1	102969.4	77.6	64.5	115.7
14	79697.7	102760.1	77.6	65.4	115.7

表 6　RGB 格式下 SMHPLC 算法与 LZ4HC 算法的比较

Level	平均比特率(kbps)		SMHPLC 的比特率或时间/LZ4HC 的比特率或时间 (%)		
	SMHPLC	LZ4HC	比特率的比	编码时间的比	解码时间的比
4	107331.0	131530.7	81.6	52.6	125.7
5	99775.8	123153.5	81.0	50.7	123.7
6	94147.6	116932.0	80.5	49.5	121.0
7	89674.4	112256.8	79.9	48.7	121.3
8	87078.4	108682.2	80.1	49.3	117.8
9	84946.4	106516.5	79.8	49.2	117.6
10	83263.8	104785.6	79.5	49.6	115.8
11	82118.7	103678.3	79.2	51.9	115.5
12	81108.6	102906.9	78.8	53.2	116.8
13	80671.6	102087.1	79.0	54.7	116.6
14	80426.4	101540.1	79.2	53.1	114.9

表 4, 表 5 中不同的搜索 Level 的 LZ4HC 算法的平均比特率都不同，从约 100 Mbps 到 130 Mbps 不等。实际上，RGB 序列 FLYG 在 Level 为 4 时的比特率是 342389.7 kbps，而 YUV 序列 CLP 在 Level 为 14 时的比特率是 55120.8 kbps。这些实验结果表明本文的方法对不同比特率的码流都有效。由于对比算法 LZ4HC，zlib 与本文方法均为无损压缩算法，所以压缩后重建图像都具有与原始图像相同的图像质量。

表 7 比较了搜索 Level 为 6 至 9 时两种色彩格式下 zlib 算法与 SMHPLC 算法之间的编码性能和复杂度。在相同 Level 下对于 YUV 和 RGB 序列，zlib 算法总体比特率分别比 SMHPLC 算法增大 2.4%~5.2%与 1.3%~5.7%，反而消耗了超过 1 倍至 2 倍的总体编码时间，总体解码时间也多出了 1 倍左右。

表 8 比较了搜索 Level 为 2 的 LZMA 与搜索 Level 为 12 时两种色彩格式下 SMHPLC 算法之间的编码性能和复杂度。对于 YUV 序列，LZMA-2 算法的总体比特率比 SMHPLC-12 算法低 10.3%，但是需要消耗更多的编解码时间，编码时间增加了 178.3%，同时解码时间增加了接近 4 倍。对于 RGB 序列，LZMA-2 算法的总体比特率只比 SMHPLC-12 算法低 6.4%，但编码时间增加了 190.7%，解码时间增加了超过 4 倍。

4　结束语

通过分析屏幕图像的特点，本文提出了基于像素串匹配的 SMHPLC 算法，在对典型屏幕图像测试序列的 Literal Length, Match Length 以及 Offset 这 3 个匹配参数进行统计和分析的基础上，提出了分类编码的思想对这些参数进行联合优化编码。此外，还引入了参数映射编码。SMHPLC 算法不但大

表 7 zlib 算法与 SMHPLC 算法的比较

色彩格式	Level	平均比特率(kbps)		zlib 的比特率或时间/ SMHPLC 的比特率或时间 (%)		
		zlib	SMHPLC	比特率的比	编码时间的比	解码时间的比
YUV	6	95765.3	92864.3	103.1	260.9	192.4
	7	93102.2	88491.3	105.2	227.1	199.0
	8	88108.7	86012.4	102.4	276.6	196.1
	9	86466.1	84013.1	102.9	326.5	198.2
RGB	6	97665.4	94147.6	103.7	272.4	198.7
	7	94813.4	89674.4	105.7	240.7	197.3
	8	88241.7	87078.4	101.3	301.9	197.4
	9	86656.1	84946.4	102.0	386.3	197.7

表 8 LZMA-2 算法与 SMHPLC-12 算法的比较

色彩	平均比特率(kbps)		LZMA-2 的比特率或时间/ SMHPLC-12 的比特率或时间 (%)		
格式	LZMA-2	SMHPLC-12	比特率的比	编码时间的比	解码时间的比
YUV	72103.0	80368.0	89.7	278.3	494.6
RGB	75927.7	81108.6	93.6	290.7	509.8

幅减少编码复杂度，而且大幅提升了压缩性能，降低了比特率。从实验结果来看，对于 YUV 序列 Level 为 13 时 SMHPLC 算法对于测试序列的压缩后码流比特率比 LZ4HC 算法总体降低了 22.4%，对于 RGB 序列 Level 为 12 时总体降低了 21.2%。SMHPLC 算法达到的对屏幕图像的压缩后码流比特率比 zlib 算法对于 YUV 和 RGB 序列分别降低了 2.4%~5.2%和 1.3%~5.7%，减少的编码时间均超过 1/2，同时解码时间也都降低了约 1/2。LZMA-2 算法虽然在比特率上比 SMHPLC-12 算法有一定的优势，但是其编解码时间却比 SMHPLC 算法长很多，特别是对于 RGB 序列编码时间长 190.7%，同时其解码时间也长了超过 4 倍，但是码流比特率只降低了 6.4%。所以 SMHPLC 算法在高性能低复杂度方面的综合优势是其他算法难以比拟的。

对 SMHPLC 算法还可以做进一步优化。在降低比特率方面，可以对匹配参数采用编码效率更高的熵编码方案，进一步提高整体编码效率；在降低算法复杂度方面，可以考虑更佳的 Chain Table 的存储和索引的方式，减少搜索时间。

参 考 文 献

[1] LIN Tao, ZHOU Kailun, and WANG Shuhui. Cloudlet-screen computing: A client-server architecture with top graphics performance[J]. *International Journal of Ad Hoc and Ubiquitous Computing*, 2013, 13(2): 96–108. doi: 10.1504/IJAHUC.2013.054174.

[2] 李德毅, 张天雷, 黄立威. 位置服务：接地气的云计算[J]. 电子学报, 2014, 42(4): 786–790. doi: 10.3969/j.issn.0372-2112.2014.04.025.
LI Deyi, ZHANG Tianlei, and HUANG Liwei. A down-to-earth cloud computing: Location-based service[J]. *Acta Electronica Sinica*, 2014, 42(4): 786–790. doi: 10.3969/j.issn.0372-2112.2014.04.025.

[3] WANG Haiyang, WANG Feng, LIU Jiangchuan, *et al*. Enabling customer-provided resources for cloud computing: Potentials, challenges, and implementation[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(7): 1874–1876. doi: 10.1109/TPDS.2014.2339841.

[4] SHIRMOHAMMADI S, ABDALLA M, AHMED D T, *et al*. Introduction to the special section on visual computing in the cloud: Cloud gaming and virtualization[J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2015, 25(12): 1955–1959. doi: 10.1109/TCSVT.2015.2473075.

[5] 张培君, 王淑慧, 周开伦, 等. 融合全色度 LZMA 与色度子采样 HEVC 的屏幕图像编码[J]. 电子与信息学报, 2013, 35(1): 196–202. doi: 10.3724/SP.J.1146.2012.00746.
ZHANG Peijun, WANG Shuhui, ZHOU Kailun, *et al*. Screen content coding by combined full-chroma LZMA and subsampled-chroma HEVC[J]. *Journal of Electronics & Information Technology*, 2013, 35(1): 196–202. doi: 10.3724/SP.J.1146.2012.00746.

[6] 陈先义, 赵利平, 林涛. 一种新的用于屏幕图像编码的 HEVC 帧内模式[J]. 电子与信息学报, 2015, 37(11): 2685–2690. doi: 10.11999/JEIT150261
CHEN Xianyi, ZHAO Liping, and LIN Tao. A new HEVC intra mode for screen content coding[J]. *Journal of Electronics & Information Technology*, 2015, 37(11): 2685–2690. doi: 10.11999/JEIT150261.

[7] LIN Tao, ZHANG Peijun, WANG Shuhui, *et al*. Mixed chroma sampling-rate high efficiency video coding for full-chroma screen content[J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2013, 23(1): 173–185. doi: 10.1109/TCSVT.2012.2223871.

[8] ZHAO Liping, LIN Tao, ZHOU Kailun, *et al*. Pseudo 2D string matching technique for high efficiency screen content coding[J]. *IEEE Transactions on Multimedia*, 2016, 18(3): 339–350. doi: 10.1109/TMM.2015.2512539.

[9] DHAWALE N. Implementation of Huffman algorithm and study for optimization[C]. International Conference on Advances in Communication and Computing Technologies (ICACACT), Mumbai, 2014: 1–6. doi: 10.1109/EIC.2015.7230711.

[10] BARTIK M, UBIK S, and KUBALIK P. LZ4 compression algorithm on FPGA[C]. IEEE International Conference on Electronics, Circuits, and Systems(ICECS), Cairo, 2015: 179–182. doi: 10.1109/ICECS.2015.7440278.

[11] ALMEIDA S, OLIVEIRA V, PINA A, *et al*. Two High-performance Alternatives to ZLIB Scientific-data

Compression. Computational Science and Its applications — ICCSA 2014[M]. Switzerland, Springer International Publishing, 2014: 623–638.

[12] SANG D K, LEE S M, SANG M L, *et al*. Compression Accelerator for Hadoop Appliance. Internet of Vehicles – Technologies and Services[M]. Switzerland, Springer International Publishing, 2014: 416–423.

[13] YANN Collet. LZ4-extremely fast compression[OL]. https://github.com/Cyan4973/lz4.git, 2016.3.

[14] YANN Collet. LZ4 Block Format Description[OL]. https://github.com/Cyan4973/lz4/lz4_Block_format.md, 2016.3.

[15] AVS 工作组文件(AVS2-P2 20110149-T-469). AVS2-P2 屏幕与混合内容视频编码(S&MCVC)通用测试条件[S]. 2016.3.
Documents of AVS2 working group. Common conditions for AVS2-P2 Screen and Mixed Content Video Coding (S&MCVC)[S]. 2016.3.

[16] ARTEM Zaytsev. LZ4-r127[OL]. https://github.com/avz/mysql-lz4.git, 2016.3 .

林　涛：男，1958 年生，教授，主要研究方向为多媒体算法和 SoC 设计.

蔡文婷：女，1990 年生，硕士生，研究方向为视频编码算法.

陈先义：男，1981 年生，博士生，研究方向为视频编码算法.

周开伦：男，1977 年生，博士生，讲师，研究方向为视频编码、屏幕图像编码、多媒体集成电路等.

王淑慧：女，1973 年生，副研究员，研究方向为视频编码、屏幕图像编码等.