Journal of Computer Applications ISSN 1001-9081 2014-05-10
计算机应用,2014,34(5):1467-1472 CODEN JYIIDU http://www.joca.cn

文章编号:1001-9081(2014)05-1467-06 doi:10.11772/j.issn.1001-9081.2014.05.1467

基于可变码长的音视频同步编码改进算法

曾 碧,林健浩*,肖 红,何元烈
(广东工业大学 计算机学院,广州 510006)
(* 通信作者电子邮箱 121492814@qq.com)

摘 要: 针对音视频同步的问题提出一种基于 H.264 帧间预测的音视频同步编码的改进算法。该算法引入可变码长的概念,将音频编码数据分为若干码组,每个码组为 2 或 3 比特的待嵌入数据。在 H.264 的帧间预测环节,可变尺寸块与码组之间根据公式确定映射关系。根据待嵌入数据来动态决定宏块分割模式的编码方式,以及根据映射关系提取数据的解码方法,使用 4×4 宏块分割模式表示嵌入数据的结束。实验结果表明,该算法使视频采集样本的峰值信噪比(PSNR)下降了 0.031 dB,码率变化率为 5.16%,产生 1.97% 的嵌入开销,但是所嵌入的音频编码数据可以正确完整地提取。因此该算法能够在增加数据嵌入容量、保持视频质量、保证数据正确性和完整性的基础上实现音视频同步编码。

关键词: H.264; 可变码长; 同步编码; 帧间预测模式; 音视频同步
中图分类号: TP391 **文献标志码:** A

Improved algorithm of audio-video synchronization coding based on variable code length

ZENG Bi, LIN Jianhao*, XIAO Hong, HE Yuanlie
(*Faculty of Computer, Guangdong University of Technology, Guangzhou Guangdong* 510006, *China*)

Abstract: To solve the synchronization problem of audio and video, an improved algorithm of audio-video synchronization coding based on H.264 inter-frame prediction was proposed. The algorithm introduced the concept of variable code length. The audio encoding data was divided into several code groups, and each code group had 2 or 3 bits of embedded data. In the stage of H.264 inter-frame prediction, the mappings between various variable size blocks and the data of code groups were based on formula. The coding method was dynamically determined for the macro block modes coding according to embedded data, and a proposed decoding method could extract the corresponding data according to the mapping relationship. Finally, the 4×4 macro block mode was used to indicate the end of the audio data. The experimental results show that the proposed algorithm enables the Peak Signal-to-Noise Ratio (PSNR) of video samples to reduce by 0.031 dB, the bit rate to increase by 5.16% and the overhead to increase by 1.97%, but the embedded audio data can be correctly and completely extracted. Therefore, the algorithm can implement the synchronization of audio and video coding while increasing the data embedding capacity, maintaining the quality of video, ensuring the correctness and completeness of the data.

Key words: H.264; variable code length; synchronization coding; inter-frame prediction mode; audio-video synchronization

0 引言

目前,多媒体信息传输系统应用非常广泛,其主要应用有视频监控、视频点播、视频会议等。在这些应用中,有一个很重要的研究内容就是音视频同步技术。

音视频同步的效果好坏直接影响到多媒体系统的服务质量(Quality of Service, QoS)的高低,音视频的不同步主要是由于视频播放时间戳与音频播放时间戳不相同导致的,其实视频与音频的播放时间差在一定范围内才会被人的感官察觉出来,当两者的时间偏差处于视频播放时间戳超前音频 90 ms 和视频播放时间戳滞后音频 20 ms 的范围时,人的感官察觉不到视频和音频的播放时间差,则这个范围是可接受的同步区域。

近年来有不少学者对基于H.264 的信息隐藏算法进行了研究[1-4],此类信息隐藏算法主要是基于 H.264 的帧内预测环节来进行信息隐藏操作的。借鉴信息隐藏算法,可将音频编码数据隐藏进视频编码数据中,进行同步编码,在解码端将视频解码的同时提取出音频编码数据,同时播放这一组音视频数据,实现音视频同步。李晓妮等[7]通过在H.264 运动估计过程利用 1/4 像素精度的运动搜索实现音视频同步编码。Qi 等[8]通过修改H.264 中适应性变动长度编码(Context Adaptive Variable Length Coding, CAVLC)的拖尾系数和非零系数来实现数据的嵌入,该方法可保持码率稳定,但预测的误差累积导致视频质量下降较大。针对视频质量的问题,有学

收稿日期:2013-11-06;修回日期:2013-12-31。 基金项目:广州市科技计划项目(2013j4300055,2013j4300033)。
作者简介:曾碧(1963-),女,广东梅州人,教授,博士研究生,CCF 高级会员,主要研究方向:嵌入式系统与智能技术、信息物理融合系统;林健浩(1988-),男,广东潮州人,硕士研究生,主要研究方向:嵌入式系统与智能技术;肖红(1972-),女,湖北咸宁人,博士研究生,主要研究方向:移动计算、物联网;何元烈(1976-),男,广东江门人,博士研究生,主要研究方向:计算机视觉、机器人、嵌入式系统。

者提出利用H.264 帧间预测环节的可变尺寸块进行数据嵌入[9-12]。针对可视电话、会议视频等的音视频同步问题，国际上也有学者提出唇同步算法[13]来解决同步问题，但存在一个嘴部自动定位的难点，算法的复杂度也比较高。传统的不基于H.264 的音视频同步方法是分别在音频和视频数据块标记时间戳[14-15]，接收端识别相同的时间戳标识进行音视频数据块的播放。也有的同步方法是生成音视频唯一的特征值并标记在相应的音视频数据块上[16]，接收端识别唯一的特征值标识进而实现音视频同步。上述这两种方案实现机制较为简单，但在网络传输环境下，一般需要使用双通道进行音视频数据传输和双缓冲区机制，而缓冲区设置较小容易溢出，设置较大则容易浪费内存资源。

文献[10]和[12]都提出一种基于H.264 帧间预测模式与数据映射的规则来进行数据的嵌入的算法，但这两种算法的数据嵌入容量小，数据嵌入量仅为平均每个宏块 2 比特。文献[11]在此基础上提出一种基于改进的 Exp-Golomb 码字结构，将可变码长分组与 7 种帧间预测映射起来进行数据嵌入，该算法将数据嵌入容量提升至平均每个宏块 2.86 比特；然而该算法无法确定数据的结束位置，也就无法准确地进行提取，则该算法还存在数据正确性和完整性的问题。本文基于文献[10-12]算法提出一种改进算法，相对于文献[10]和[12]来说，提升了数据嵌入容量，也解决文献[11]的数据正确性和完整性问题。

1 基于可变尺寸块同步编码算法分析

H.264 采用不同大小和形状的宏块分割与亚分割方法，一个宏块按照 16×16、16×8、8×16 或 8×8 进行分割，如图 1(a)所示，如果选择了 8×8 分割，那么还可以按照 8×8、8×4、4×8 或 4×4 来进行亚宏块分割，如图 1(b)所示。所以，宏块分割与亚宏块分割的方法使得每个宏块中包含有许多大小不同的块，利用各种大小和形状不同的块来进行运动补偿。

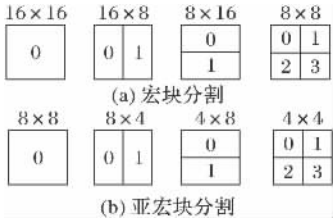

图 1 H.264 帧间预测宏块与亚宏块分割模式

文献[10]和[12]所提供的算法，使用可变尺寸块 7 种模式中的 4 种，每一个宏块表示 2 比特的数据，分别是 00、01、10 和 11。

文献[11]采用信息可变长分组方法来进行信息隐藏，理论上每个宏块可以嵌入 2.86 比特的信息，文献[11]算法的映射规则如表 1 所示，其中 *Code* 表示帧间预测模式的十进制数值，*M* 表示数据的长度，*INFO* 表示数据的十进制数值。文献[11]算法使用所有 7 种模式，当待隐藏的信息已经结束时，却没办法表示信息结束位置；当隐藏的信息末尾还剩有二进制信息为 00、01、10、0 和 1 时，无法进行信息的嵌入和提取，影响了信息的正确性和完整性。

本文对文献[11]和[12]的算法进行改进，只使用帧间预测的 6 种模式来进行音频数据嵌入，6 种模式分别为 16×16、16×8、8×16、8×8、8×4 和 4×8，使用 4×4 模式用来标识数据的结束。这样既能够保证数据的正确性和完整性，又能保持平均每个宏块嵌入 2.67 比特的数据嵌入容量，比文献[10]和[12]嵌入的数据量多，同时也解决文献[11]的正确性和完整性问题。

表 1 改进的 Exp-Golomb 的码字结构及块类型映射规则

预测模式	*Code*	*M*	*INFO*	信息组	预测模式	*Code*	*M*	*INFO*	信息组
16×16	0	2	3	11	8×4	4	3	3	011
16×8	1	3	0	000	4×8	5	3	4	100
8×16	2	3	1	001	4×4	6	3	5	101
8×8	3	3	2	010					

2 同步编码算法实现

2.1 可变码长分组

H.264 帧间预测阶段的可变尺寸块总共有 7 种：16×16、16×8、8×16、8×8、8×4、4×8 和 4×4。其中大的宏块分割主要用于视频图像中较为平缓的区域，例如背景颜色一样的区域；而小的宏块分割主要用于表现细节，例如人的轮廓。因此，4×4 模式使用的频率是最少的，用该模式来标识数据的结束位置和表示剩余数据是最合适的。

本文只使用前 6 种来进行数据表示，输入音频编码数据，然后将二进制数据分为若干 2 或 3 比特的可变分组。其中有一部分可变尺寸块用来表示 2 比特的数据分组，有一部分可变尺寸块表示 3 比特的数据分组。根据文献[11]所提出的公式，本文算法对其进行修改如下：

$$Len = Floor(\mathrm{lb}(Code + 6)) \quad (1)$$

$$Info = Code + 6 - 2^{Len} \quad (2)$$

Len 表示二进制音频编码数据分组的比特数，*Info* 表示数据分组的十进制数值，*Code* 表示帧间预测的模式序号，*Floor* 表示向下取整。

表 2 数据分组与预测模式的映射关系

预测模式	*Code*	*Len*	*Info*	信息组	预测模式	*Code*	*Len*	*Info*	信息组
16×16	0	2	2	10	8×8	3	3	1	001
16×8	1	2	3	11	8×4	4	3	2	010
8×16	2	3	0	000	4×8	5	3	3	011

表 2 所示的映射关系是根据式(1)、(2)得出的 *Code* 预设的模式顺序，16×16 的 *Code* 为 0，那么根据式(1)可得 $Len = Floor(\mathrm{lb}(0 + 6))$，结果 *Len* 为 2，然后代入式(2)得出 *Info* 为 2，表示为二进制数据即为 10。同样的，其他的数据分组也根据式(1)、(2)得出。

剩余的帧间预测 4×4 模式在本文的算法中有两个作用：第一，当音频编码数据嵌入完成时，用 2 个 4×4 模式来标识数据的结束；第二，当数据分组剩余二进制数据为 00、01、0 或 1 时，采用一个 4×4 模式后面紧接着 16×16、16×8、8×16 或

8×8 模式的组合来表示。

2.2　音频编码数据嵌入

2.2.1　数据分组

根据式(2),可以得出式(3):

$$Code = Info - 6 + 2^{Len} \tag{3}$$

将音频编码压缩后的数据表现为二进制数值,每次读取 2 或 3 比特数据,直至数据结束。二进制和十进制数据数值分别表示为$(x)_2$和$(x)_{10}$,例如$(10)_2$、$(10)_{10}$分别表示为二进制的 10 和十进制的 10。分组具体流程如下:

1) 首先读入 3 比特二进制数据(binary Data, binData),假如数据大于$(011)_2$,那么进入 2);假如数据不大于$(011)_2$,那么将 $Len = 3$ 和 $Info$ 数值代入式(3),得出 $Code$ 数值,进入 3)。

2) 读入 2 比特二进制数据,将 $Len = 2$ 和 $Info$ 数值代入式(3),得出 $Code$ 数值,进入 3)。

3) 根据计算得到的 $Code$,从表 2 中可得出对应的帧间预测模式,返回 1)。

图 2 给出一个例子来说明数据分组对应的预测模式,第一组二进制数据$(101)_2$比$(011)_2$大,那么改为读取 2 比特数据$(10)_2$,即 $Len = 2$;另外二进制数值$(10)_2$等于十进制数值$(2)_{10}$,则 $Info = 2$。把 Len 和 $Info$ 代入式(3)为 $Code = 2 - 6 + 2^2$,得 $Code = 0$,从表 2 中得$(10)_2$对应的预测模式为 16×16。

得到第一组 2 比特数据对应的预测模式之后,读取数据位置向前位移 Len 个位置,读取第二组数据,因为$(110)_2$比$(011)_2$大,也改为读取 2 比特数据$(11)_2$,数据代入式(3)为 $Code = 3 - 6 + 2^2$,得 $Code = 1$,对应预测模式为 16×8。

读取位置继续向前位移 Len 个位置,依此类推,得出其他的分组数据对应的预测模式。

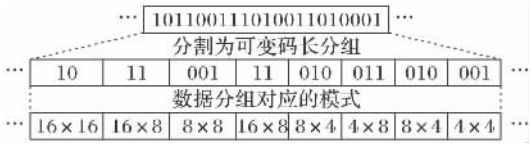

图 2　数据分组与对应的预测模式

2.2.2　数据嵌入

H.264 中引入了 B_Skip 类型宏块和 P_Skip 类型宏块,其中:

1) B_Skip 类型宏块表示像素残差和运动矢量残差都为 0,那么解码端会通过 Direct 预测模式计算出前向和后向运动矢量 MV,直接利用前后向运动矢量来得到像素预测值;

2) P_Skip 类型宏块表示宏块已经没有更多的数据了,像素残差和无运动矢量残差也都为 0。

H.26 在图像平坦的区域采用 Skip 类型宏块,而 Skip 类型宏块没有携带任何数据,编码端不需要传送运动矢量,解码端根据已重建的周围宏块来构建 Skip 宏块,因此 Skip 类型宏块不适合用来进行数据嵌入。

假如在数据嵌入过程中遇到 Skip 类型宏块,则不使用该宏块来嵌入音频数据,具体的流程如下描述和图 3 所示:

1) 执行 2.2.1 节的数据分组流程,得到读取的二进制数据长度 Len,待嵌入数据对应的预测模式(insert Mode, $insertM$),进入 2)。

2) 获取当前视频编码宏块信息,判断如果当前宏块属于 $Skip$ 类型,那么进入 3);如果不属于,则根据 2.2.1 节得到的预测模式 $insertM$,将 $insertM$ 作为当前帧间预测模式(current mode, $currentM$),进入 4)。

3) 获取下一视频编码宏块信息,重新返回 2)。

4) 将当前数据位置(current Position, $currentPos$)位移 Len 个位置,判断 $currentPos$ 是否为结束位置,如果是,则进入数据结束处理;不是,则返回 1)。

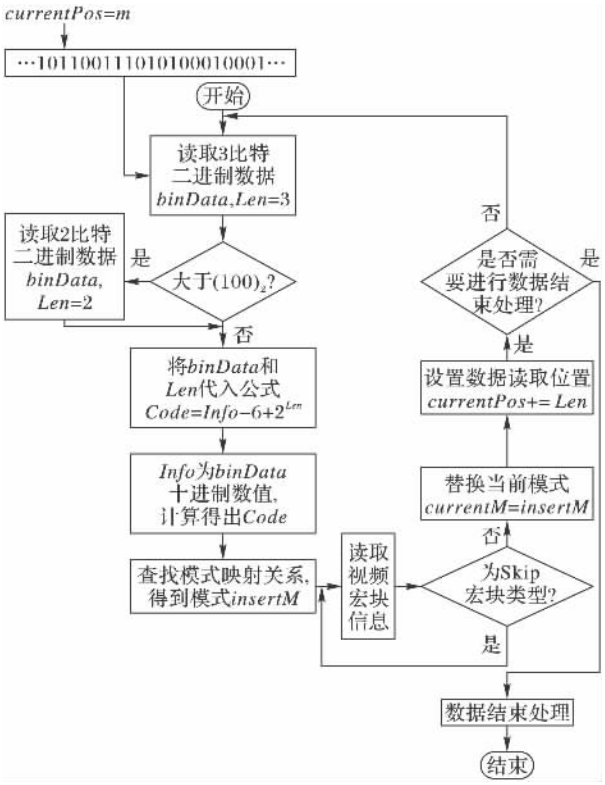

图 3　数据嵌入流程

2.2.3　数据结束处理

2.2.2 节中描述了数据嵌入的过程,当音频编码数据读取完毕或剩余部分无法表示的数据时,则应该进行数据结束处理,具体的数据结束处理过程(如图 4)如下:

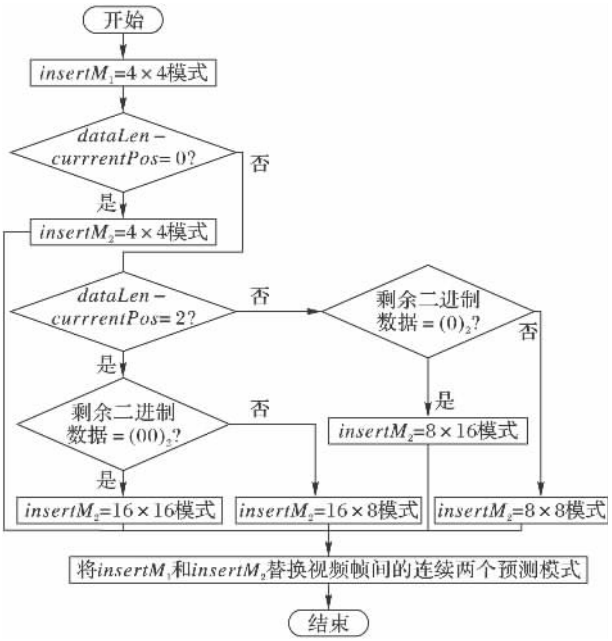

图 4　数据结束处理流程

1) 如果数据读取完毕,则使用 2 个 4×4 模式来进行帧间

预测模式的替换;如果数据未读取完毕,则进入 2)。

2) 如果剩余数据分组为$(00)_2$,使用 4×4 模式和 16×16 模式来表示;否则进入 3)。

3) 如果剩余数据分组为$(01)_2$,使用 4×4 模式和 16×8 模式来表示;否则进入 4)。

4) 如果剩余数据分组为$(0)_2$,使用 4×4 模式和 8×16 模式来表示;否则使用 4×4 模式和 8×8 模式来表示$(1)_2$。

图 4 中 $insertM_1$ 和 $insertM_2$ 为预测模式,$currentPos$ 为数据读取位置,$dataLen$ 为数据的总长度,$(00)_2$ 表示 2 比特的二进制数值 00,$(0)_2$ 表示 1 比特的二进制数值 0。

2.3 音频编码数据提取

在解码端进行数据提取步骤如下,其中: $remainingData$ 表示为剩余的数据,$endFlag$ 表示为结束标志位。

1) 获取宏块信息,判断当前宏块是否为 4×4 预测模式,如果是则进入 2);如果不是,则进行 3)。

2) 接收下一宏块信息,进行一系列判断,提取剩余二进制数据,下面是提取剩余二进制数据的伪代码过程:

```
switch(当前宏块模式)
{
  case 16×16: remainingData = (00)2; break;
  case 16×8: remainingData = (01)2; break;
  case 8×16: remainingData = (0)2; break;
  case 8×8: remainingData = (1)2; break;
  case 4×4: break;
}
```

提取剩余数据完成之后,设置标志位 $endFlag$ = True,结束数据提取过程。

3) 进行数据提取,提取伪代码如下:

```
switch(当前宏块模式)
{
  case 16×16: binData = (10)2; break;
  case 16×8: binData = (11)2; break;
  case 8×16: binData = (000)2; break;
  case 8×8: binData = (001)2; break;
  case 8×4: binData = (010)2; break;
  case 4×8: binData = (011)2; break;
}
```

提取数据完成之后,返回 1)。

3 实验结果分析

为了验证本文改进算法的可行性,使用H.264/AVC 实验平台软件 JM8.6 进行实验,本实验使用了 16 组 QCIF(Quarter Common Intermediate Format)格式的视频测试序列: akiyo、bridge-far、carphone、city、crew、football、foreman、hall、harbour、highway、ice、miss-america、mobile、mother-daughter、news 和 soccer。

首先,分别对这 16 组视频测试序列(每组 100 帧)嵌入 1 600比特的音频编码数据,得出实验数据。并与文献[10-12]在相同的实验条件下进行实验,对比实验数据。H.264/AVC 实验平台软件 JM8.6 编码端的主要参数配置如表 3 所示。

表 3 JM8.6 编码端的参数配置

参数名	值	参数名	值
Profile	Baseline	Sequence Type	IBPBP
Frames	100	Rate Distortion Optimization	used
Frame Rate	30 fps	8×8SubBlocks	used
Number Reference Frames	5	Rate Control	Disabled

实验结果数据从亮度分量 Y 的信噪比(Peak Signal-to-Noise Ratio Y, PSNR Y)、码率 $BitRate$ 和算法的额外开销 O_e 三类数据进行处理。根据文献[12]对额外开销 O_e 的描述如下:

$$O_e = \frac{H_e - O_v - A_e}{O_v + A_e} \times 100\% \tag{4}$$

其中: O_e 为嵌入音频数据的额外开销,O_v 和 A_e 分别为视频和音频各自编码所产生的数据量,H_e 为嵌入音频编码数据后的视频编码所产生的数据量。

对 100 帧视频测试序列嵌入 1 600 比特的音频编码数据,分别得出 16 组数据,如表 4 所示。其中,$PSNR\ Y$ 列中的负值表示使用本文嵌入算法比 H.264/AVC 参考软件算法在亮度分量 Y 信噪比的下降数,正值则表示增加数; $BitRate$ 列表示嵌入算法相比原标准算法的比特率变化情况; O_e 列表示使用嵌入算法比原标准算法额外开销。

表 4 本文算法相比 H.264/AVC 算法实验数据变化量

视频序列	$PSNR\ Y$/dB	$BitRate$/%	O_e/%	视频序列	$PSNR\ Y$/dB	$BitRate$/%	O_e/%
akiyo	0.06	2.35	1.89	harbour	-0.017	1.11	0.99
bridge-far	0.018	3.52	2.65	highway	0.017	2.91	2.42
carphone	0.003	2.01	1.68	ice	-0.042	3.07	2.83
city	-0.004	1.80	1.57	miss-america	0.011	3.23	2.53
crew	-0.003	1.26	1.11	mobile	-0.006	1.04	0.94
football	-0.035	1.85	1.76	mother-daughter	0.023	2.31	1.84
foreman	-0.018	1.92	1.65	news	0.009	1.43	1.15
hall	0.021	1.53	1.20	soccer	-0.022	2.29	2.12

从表 4 显示的数据结果可以看出,$PSNR\ Y$ 下降相对较多的有 3 个视频序列,分别为 football、ice 和 soccer 视频序列,说明对于运动剧烈的视频序列,嵌入算法引起的 $PSNR\ Y$ 下降比较明显,对于运动平缓的视频序列,视频序列的 $PSNR\ Y$ 会平稳一些,甚至有的视频序列呈现 $PSNR\ Y$ 增加的现象,例如: akiyo、bridge-far、carphone 等。

下面将对比一下 football、ice 和 soccer 原始视频测试序列、未嵌入音频编码数据解码后及嵌入音频编码数据解码后的第 10 帧图像,如图 5~7 所示,对比这 3 组视频图像表明,感官上嵌入数据后的视频质量不会出现严重失真。

本文算法与文献[10]的数据隐藏算法进行实验数据对比如图 8 和 9 所示,具体数据对比如表 5 所示。对比视频序列(QCIF 格式)的 $PSNR\ Y$,明显本文算法的性能要比文献[10]算法好;再对比视频序列的码率 $BitRate$,文献[10]算法使得视频序列 D(foreman)的 $BitRate$ 的变化量较大,而本文算法则比较平稳,除此之外,本文算法与文献[10]算法基本持平。

图 5　football、ice 和 soccer 原始视频图像

图 6　football、ice 和 soccer 未嵌入数据的解码图像

图 7　football、ice 和 soccer 嵌入数据后的解码图像

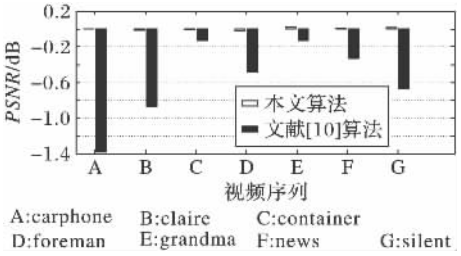

图 8　本文算法与文献 [10] 算法的 PSNR Y 值

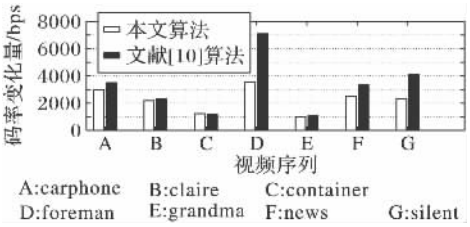

图 9　本文算法与文献 [10] 算法的 BitRate 值

从表 6 的实验数据表明，虽然在视频序列的 *PSNR Y* 方面，本文算法总体上略优于文献 [11] 算法，但在 *BitRate* 方面，本文算法却比文献 [11] 算法略差。最重要的一点，本文算法使用 6 种预测模式来进行数据嵌入操作，保留一种预测模式来解决文献 [11] 存在的数据正确性和完整性问题。

文献 [12] 在实验中采用的视频测试序列是 CIF(Common Intermediate Format) 格式，如表 7 所示，本文算法使用 stefan、mobile 和 tempete 三个视频测试序列进行实验。从 *PSNR Y* 和 O_e 两方面进行对比，本文算法在 stefan 视频序列的数据表明性能略低于文献 [12] 算法，但其他两个视频序列的性能却优于文献 [12] 算法; 从数据嵌入量方面对比，本文算法高于文献 [12] 算法。

表 5　本文算法相比文献 [10] 算法实验数据变化量

视频序列	本文算法		文献 [10] 算法	
	PSNR Y /dB	*BitRate* /%	*PSNR Y* /dB	*BitRate* /%
carphone(QCIF)	0.003	2.01	−1.380	2.38
claire(QCIF)	−0.012	2.85	−0.870	3.05
container(QCIF)	−0.010	1.42	−0.130	0.78
foreman(QCIF)	−0.018	1.92	−0.480	3.87
grandma(QCIF)	0.019	1.97	−0.130	0.82
news(QCIF)	0.009	1.43	−0.330	1.93
silent(QCIF)	0.017	1.30	−0.670	2.31

表 6　本文算法相比文献 [11] 算法实验数据变化量

视频序列	本文算法		文献 [11] 算法	
	PSNR Y /dB	*BitRate* /%	*PSNR Y* /dB	*BitRate* /%
carphone(QCIF)	0.003	2.01	−0.025	0.34
foreman(QCIF)	−0.018	1.92	−0.020	0.09
grandma(QCIF)	0.019	1.97	0.020	3.23
highway(QCIF)	0.017	2.91	−0.010	0.43
mobile(QCIF)	−0.006	1.04	−0.005	0.89
news(QCIF)	0.009	1.43	0.011	1.60
silent(QCIF)	0.017	1.30	0.011	0.25

表 7　本文算法相比文献 [12] 算法实验数据变化量

视频序列	本文算法		文献 [12] 算法	
	PSNR Y /dB	O_e /%	*PSNR Y* /dB	O_e /%
stefan(CIF)	−0.024	4.61	−0.017	3.36
tempete(CIF)	−0.001	1.27	−0.022	1.73
mobile(CIF)	−0.001	1.36	−0.027	1.64

图 10，11 和 12 分别为 stefan、tempete 和 mobile 原始视频测试序列、未嵌入音频编码数据解码后及嵌入音频编码数据解码后的第 20 帧图像，对比这 3 组视频图像表明，未能察觉明显的视频失真问题。

图 10　stefan、mobile 和 tempete 原始视频图像

图 11　stefan、mobile 和 tempete 未嵌入数据后的解码图像

图 12　stefan、mobile 和 tempete 嵌入数据后的解码图像

为了更进一步进行实际验证，采集了 100 帧样本视频序列(320 × 240 像素) ，嵌入数据为 3.3 s、8 kHz、16 位单声道的音频数据经过 G.729 标准压缩后的编码数据，得到的原视频序列的 *PSNR Y* 和嵌入数据后视频序列的 *PSNR Y*，具体数据对比如图 13 所示。

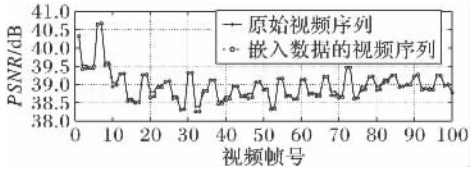

图 13　对比样本视频嵌入数据前后 *PSNR Y* 变化

从图 13 看出，嵌入数据后的视频序列 *PSNR Y* 变化较小，没有太大的波动，两条数据曲线基本一致。图 14 为 *Sample Video* 原始样本视频序列、未嵌入数据的解码图像和嵌入数据解码后的第 30 帧图像。从感官角度来对比，嵌入数据后的解码图像与原始解码图像并无太大差异，无明显的失真问题; 从数据角度来看，嵌入数据前后视频序列的统计数据

(*EncodeData* 为数据量) 如表 8 所示,峰值信噪比 *PSNR Y* 下降了 0.031,码率 *BitRate* 增加 5.16%,嵌入开销 O_e 增加 1.97%,这三个数据的变化情况都比较理想,处于可接受范围。

(a) 原始图像 (b) 未嵌入数据解码图像 (c) 嵌入数据后解码图像

图 14 原始视频图像和嵌入数据前后的解码图像

表 8 嵌入数据前后的视频序列统计数据

视频序列	*PSNR Y* /dB	*BitRate* /bps	*EncodeData* /B
原始样本视频	38.987	285.55	120 167
嵌入数据的视频	38.956	300.28	124 575

4 结语

本文提出一种基于H.264 帧间预测的可变尺寸块嵌入音频编码数据的改进算法。该算法使用 7 种帧间预测模式中的 6 种模式来表示音频数据,利用模式组合来标识数据结束位置和表示剩余的数据。

本文算法将音频编码数据根据公式分为 2 或 3 比特的可变码长分组数据,采用 6 种模式来进行表示,平均每个宏块能嵌入 2.67 比特的数据量,嵌入数据容量大大提高,在保持嵌入数据容量较高的同时,还能保证音频数据被正确且完整地提取出来,实现音视频同步。

参考文献:

[1] YIN Q, WANG H, ZHAO Y. An information hiding algorithm based on intra-prediction modes for H.264 video stream [J]. Journal of Optoelectronics · Laser, 2012, 23(11): 2194 – 2199. (尹秋来,王宏霞,赵杨. 一种基于预测模式的 H.264 视频信息隐藏算法[J]. 光电子·激光, 2012, 23(11): 2194 – 2199.)

[2] XU D, WANG R. An improved information hiding algorithm with prediction mode modulating for H.264/AVC [J]. Opto-Electronic Engineering, 2011, 38(11): 93 – 99. (徐达文,王让定. 一种基于预测模式的H.264/AVC 视频信息隐藏改进算法[J]. 光电工程, 2011, 38(11): 93 – 99.)

[3] ZHENG J, TANG J, GUO L. H.264/AVC compressed domain data hiding algorithm based on fast recoding [J]. Journal of University of Science and Technology of China, 2013, 43(1): 35 – 41. (郑建峰,唐建,郭立. 一种基于快速重编码的H.264/AVC 压缩域信息隐藏方案[J]. 中国科学技术大学学报, 2013, 43(1): 35 – 41.)

[4] ZHANG J. Research on key technology of video information hiding [D]. Hefei: University of Science and Technology of China, 2011. (张家骥. 视频图像中信息隐藏的关键技术研究[D]. 合肥: 中国科学技术大学, 2011.)

[5] HU Y, ZHANG C, SU Y. Information hiding for H.264/AVC [J]. Acta Electronica Sinica, 2008, 36(4): 690 – 694. (胡洋,张春田,苏育挺. 基于H.264/AVC 的视频信息隐藏算法[J]. 电子学报, 2008, 36(4): 690 – 694.)

[6] XIAO C, WANG S, SI W. High bitrate information hiding technique for video in video [J]. Journal of Beijing Polytechnic University, 2011, 37(8): 1249 – 1254. (肖创柏,王首道,司薇. 高比特率信息隐藏技术的视频嵌入视频方案[J]. 北京工业大学学报, 2011, 37(8): 1249 – 1254.)

[7] LI X, CHEN H, CHEN M, *et al*. Audio-video synchronous coding based on motion estimation in H.264 [J]. Journal of Jilin University: Engineering and Technology Edition, 2012, 42(5): 1321 – 1326. (李晓妮,陈贺新,陈绵书,等. 基于H.264 运动估计的音视频同步编码技术[J]. 吉林大学学报: 工学版, 2012, 42(5): 1321 – 1326.)

[8] QI X, CHEN M, CHEN H. A CAVLC embedded method for audio-video synchronization coding based on H.264 [C]// Proceedings of the 2011 International Conference on Multimedia Technology. Piscataway: IEEE Press, 2011: 16 – 19.

[9] LIU T. Research on audio-video synchronization coding based on inter prediction blocks of varying size for H.264 [D]. Changchun: Jilin University, 2011. (刘添. 基于H.264 可变尺寸帧间预测的音视频同步算法研究[D]. 长春: 吉林大学, 2011.)

[10] KAPOTAS S K, VARSAKI E E, SKODRAS A N. Data hiding in H.264 encoded video sequences [C]// Proceedings of the 2007 IEEE 9th Workshop on Multimedia Signal Processing. Piscataway: IEEE Press, 2007: 373 – 376.

[11] XUAN M, JIANG J. H.264 information hiding algorithm with high capacity based on variable length group structure [J]. Journal of Circuits and Systems, 2012, 17(6): 42 – 48. (宣曼,蒋建国. 基于变长分组结构的H.264 大容量信息隐藏算法[J]. 电路与系统学报, 2012, 17(6): 42 – 48.)

[12] LI X, CHEN H, SUN Y, *et al*. Embedded audio-video synchronization coding based on H.264 [J]. Journal of Jilin University: Engineering and Technology Edition, 2011, 41(5): 1475 – 1479. (李晓妮,陈贺新,孙元,等. 基于H.264 的嵌入式音视频同步编码技术[J]. 吉林大学学报: 工学版, 2011, 41(5): 1475 – 1479.)

[13] AGGARWAL S, JINDAL A. Comprehensive overview of various lip synchronization techniques [C]// Proceedings of the 2008 IEEE International Symposium on Biometrics and Security Technologies. Piscataway: IEEE Press, 2008: 1 – 6.

[14] ZHANG J, LI Y, WEI Y. Using timestamp to realize audio-video synchronization in real-time streaming media transmission [C]// Proceedings of the 2008 International Conference on Audio, Language and Image Processing. Piscataway: IEEE Press, 2008: 1073 – 1076.

[15] EL-HELALY M, AMER A. Synchronization of processed audio-video signals using time-stamps [C]// Proceedings of the 2007 IEEE International Conference on Image Processing. Piscataway: IEEE Press, 2007, 6: 193 – 196.

[16] RADHAKRISHNAN R, TERRY K, BAUER C. Audio and video signatures for synchronization [C]// Proceedings of the 2008 IEEE International Conference on Multimedia and Expo. Piscataway: IEEE Press, 2008: 1549 – 1552.