一种面向多投影显示的全景视频分块编码方法

梅元乔[1] 姜忠鼎[2]

[1](复旦大学上海市数据科学重点实验室 上海 201203)

[2](复旦大学软件学院 上海 201203)

摘 要 近年来虚拟现实技术快速发展,全景视频内容生成技术不断进步。内容呈现方面,多投影显示设备可通过集群驱动解决单机性能有限的问题,具有视角广、分辨率高、支持多人参与的优点。针对多投影显示设备的特点,设计并实现了一套全景视频分块编码格式及其相应的转码工具。该格式以现有的 JPEG 图像序列编码研究为基础,添加对立体全景视频分块编码的支持,实现双分辨率视频流动态适配机制,保证视角变化时视频的流畅播放,并添加非均匀分块编码的支持以针对实际需求定制分块参数,优化播放性能。实验结果表明,该方法可以有效提升多投影显示设备中高分辨率全景视频播放的流畅度。

关键词 虚拟现实 全景视频 多投影显示 视频编解码

中图分类号 TP3 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.03.025

A TILE-BASED PANORAMIC VIDEO CODING METHOD FOR MULTI-PROJECTOR DISPLAY

Mei Yuanqiao[1] Jiang Zhongding[2]

[1](*Shanghai Key Laboratory of Data Science*, *Fudan University*, *Shanghai 201203*, *China*)

[2](*Software School*, *Fudan University*, *Shanghai 201203*, *China*)

Abstract With the fast development of virtual reality in recent years, panoramic video generation technology has made continuous progress. In the aspect of content presentation, multi projection display device can solve the problem of the limited performance of a single machine through cluster drive, which has the advantages of wide view angle, high resolution and support for multi-user participation. According to the characteristics of multi-projector display, we designed and implemented a set of tile-based panoramic video coding formats and their corresponding transcoding tools. Based on the existing JPEG image sequence coding research, this format added support for stereo tile-based panoramic video coding, and realized the dynamic adaptation mechanism of dual-resolution video stream to ensure smooth playback of video when the perspective changed. In addition, support for non-uniform block coding was added to customize block parameters for actual needs and optimize playback performance. The experiments show that our method efficiently improves the fluency of high-definition panoramic video playback in multi-projector display devices.

Keywords Virtual reality Panoramic video Multi-projector display Video codec

0 引 言

虚拟现实被称为“下一代通用计算平台”,它使用计算机创建出逼真的三维立体虚拟场景,用户可以与虚拟环境进行交互,从而得到强烈的沉浸感。现有的虚拟现实呈现设备主要包括头戴式显示设备和多投影显示设备。头戴式显示设备在用户左右眼前的屏幕中显示带有视差的立体图像,并根据用户的头部运动实时呈现相应视角的画面内容。头戴式显示设备的缺点在于只能单个用户佩戴,难以多人参与,并且一般为单机驱动,容易出现性能瓶颈。多投影显示设备使用

收稿日期:2019-04-15。国家自然科学基金项目(60803064)。**梅元乔**,硕士生,主研领域:计算机图形学,虚拟现实,人机交互。**姜忠鼎**,副教授。

LCD 拼接屏和投影机阵列构建大尺寸的显示屏幕,可通过集群驱动解决单机性能有限的问题,具有视角广、分辨率高、支持多人参与的优点。

全景视频观看是虚拟现实的重要应用场景,其将全空间的三维场景映射到二维空间中,用户无需搭建三维场景即可体验身临其境的沉浸感。目前超高清全景视频拍摄及生成技术快速发展,Insta360 最新 VR 相机 Titan 已支持 11K 2D 及 10K 3D 全景视频的拍摄。超高分辨率也对视频编码和传输技术提出了挑战,传统的编码方式已不再能满足 VR 全景视频超高分辨率的需求。大部分 VR 显示设备的呈现区域有限,因此全景视频中大量内容无需显示,针对这一特点,研究者使用分块编码[1-3]和 FOV 传输[4-6]的策略节约视频解码和网络传输的开销,从而实现超高分辨率全景视频的流畅播放。

新一代的视频编码标准 HEVC 提供了 tile 机制,允许将视频按照横向和纵向划分成多块,每个分块可独立解码。Misra 等[2]从并行化和 MTU 两个方面详细讨论了如何利用 HEVC 中的分块机制提高解码和传输效率。应用实例方面,许多研究者基于 HEVC 的分块机制实现了全景视频的分块编码和传输系统。Skupin 等[3]实现了面向 VR 头盔的全景视频的视口自适应编解码策略,为视口区域提供高分辨率码流,非视口区域则提供低分辨率的码流。Zare 等[4]使用分块的策略优化全景视频的传输效率,为每个视频准备两个不同码率的版本,播放时用户 FOV 范围内的图像以高码率传输,FOV 外的图像则以低码率传输。Corbillon 等[5]扩展了上述方法,设计了一种视口自适应的全景视频流式传输系统,每个视频的多个版本不仅码率不同,还在不同区域的质量上存在差异,视频的部分区域质量高于该视频的其他区域,客户端播放时选择码率合适且能为其视口区域提供全质量呈现的视频版本。

但是现有的研究还存在一些问题。视频每帧内容从解码到显示需要一段时间,很可能解码和显示时的可见分块列表不相同,导致某些需要呈现的分块尚无内容。为此许多研究者对视角运动轨迹进行预测,以提高分块可见性检测的准确率,如 Fan 等[7]利用头部运动信息和视频内容构建神经网络进行预测。然而可见分块预测的准确率无法达到 100%。针对该问题,现有的方案[3-5]基本都采用低分辨率图像的方案处理。头戴式显示设备中仅单人观看视频,视线集中于视口中央区域,但是多投影显示设备支持多人参与,因此屏幕边缘区域呈现内容不清晰将严重影响用户体验。

HEVC 对视频帧随机访问的支持较弱,视频跳转到某个需要补偿的非关键帧分块时需要较长的解码时间。此外,多投影显示通过集群驱动以及拼接融合技术可达到很高的分辨率,而 HEVC 视频的分辨率上限只有 8K。针对这些问题,已有的研究实现了一种基于 JPEG 图像序列的视频帧切分编码格式[11],每帧均可独立解码,以一定的压缩率为代价,降低了分块视频的随机访问延迟,从而缩短分块预测错误时补偿解码的等待时间。此外 JPEG 分块编码不存在分辨率上限,可以满足多投影显示的需求。然而视频播放过程中需要补偿解码的分块数量过多时,依然会造成播放卡顿。

基于上述讨论,本文实现了一种面向多投影显示的全景视频分块编码方法,减少单机的性能开销。实验结果表明,该分块编码格式可以有效地提升多投影显示设备中高分辨率全景视频播放的流畅度。

1 基于 JPEG 图像序列的全景视频分块编码

全景视频有许多不同的映射方式[9-10],目前最主流的是等量矩形投影(Equirectangular Projection,ERP),也被称为经纬图。本文的分块编码方法针对 ERP 全景视频。

传统的视频编码方式每帧以整幅图片为单位进行编码,但是对于全景视频而言,由于显示设备的呈现范围有限,每帧图像中大量区域无需显示,却依然占用了解码和渲染的计算资源,导致播放性能降低。将视频每帧的图像数据分块编码,播放时仅解码和渲染可见分块是解决该问题的基本思想。如视频共有 N 个分块,所有分块交集为空,并集为视频全部内容,第 i 个分块包含的数据量为 S_i,则视频每帧所需处理的总数据量 T 如下:

$$f(i)=\begin{cases}S_i & visible(i)\\ 0 & \text{其他}\end{cases} \tag{1}$$

$$T=\sum_{i=1}^{N}f(i) \tag{2}$$

1.1 立体全景视频分块编码

本文的全景视频分块编码方法提供了对左右立体、上下立体和左右分离三种立体格式的支持,编码时左右眼画面均采用完全相同的分块方式。对左右立体和上下立体而言,每个分块由其左右眼的画面按照左右或上下排列的方式拼合而成,每个分块的画面依次

存储。对左右分离格式而言,左眼和右眼的每个分块单独编码,存储时首先依次写入左眼所有分块,然后依次写入右眼所有分块,如图 1 所示。

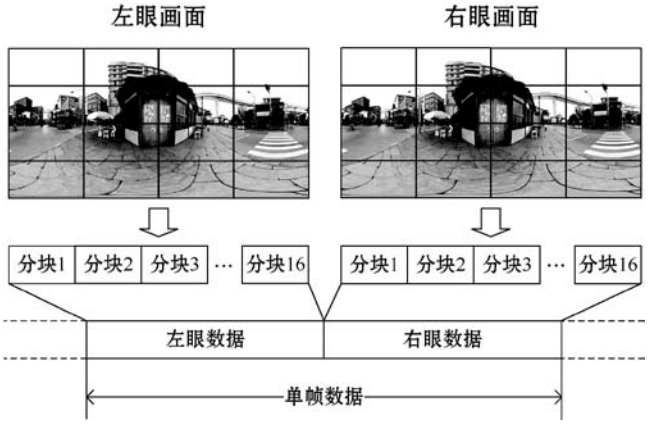


图 1　左右分离立体全景视频分块编码示意图

立体显示方面常见的技术包括被动立体和主动立体。被动立体的多投影显示系统中,单机仅需解码和渲染单眼画面,而左右分离的全景视频仅需读取和解码单眼的分块数据,可以显著提升多投影显示系统的播放效率。对于主动立体的多投影显示系统,则可使用左右立体或上下立体减少需要解码的分块数量。

1.2　双分辨率视频流动态适配

全景视频观看视角旋转时,可见分块列表实时变化。虽然 JPEG 图像序列降低了分块视频的随机访问延迟,但是当需要补偿解码的分块数量过多时,依然会造成播放卡顿。因此本文的分块编码格式添加了双分辨率流适配机制:每帧数据除了包含所有分块的原分辨率图像,还包含一幅未分块的低分辨率图像,存储于每帧分块数据的末尾,如图 2 所示。

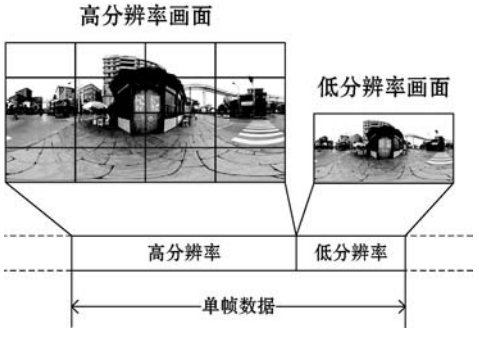


图 2　双分辨率视频流示意图

播放器解码预测可见的分块图像以及该幅低分辨率图像。主线程每帧渲染之前获取需要补偿解码的分块列表,然后使用预测算法计算补偿解码的总时间,如预测时间小于一定阈值则进行补偿解码,否则在未解码的区域呈现低分辨率图像,避免当前帧卡顿。该机制可以在不明显降低视频播放流畅度的同时,为用户尽可能提供高分辨率的视频内容。

1.3　非均匀分块编码

一般而言,较小的分块可以减少视口相同时所需解码和渲染的数据量,但是分块数量过多也会造成额外的指令开销。本文的分块编码格式在已有的均匀分块基础上,添加了非均匀分块的支持。在某些实际应用场景下,如视频极点附近存在重要内容,或者多投影显示的屏幕参数与视频边界不匹配时,非均匀分块可以带来更好的性能。而在其他情况下,非均匀分块与均匀分块的播放性能接近。

本文的分块编码方法支持横向和纵向的非均匀划分。如视频总分辨率为 $W \times H$,横向按照 $(x_1, x_2, \cdots, x_m)$ 的比例划分成 M 块,纵向按照 $(y_1, y_2, \cdots, y_n)$ 的比例划分成 N 块,bpp 为每个像素所占的字节数,则第 i 行第 j 列的分块包含的数据量 $S_{i,j}$ 如下:

$$W_j = \left(\frac{x_j}{\sum_{k=1}^{M} x_k} \times W \right) \tag{3}$$

$$H_i = \left(\frac{y_i}{\sum_{k=1}^{N} y_k} \times H \right) \tag{4}$$

$$S_{i,j} = W_j \times H_i \times bpp \tag{5}$$

2　分块参数确定流程

经本文研究,多投影显示系统中全景视频分块编码的参数需综合考虑以下因素确定:单机视口范围、屏幕尺寸、视频内容、解码和渲染指令本身的开销。下面给出分块参数确定的整体流程:

(1) 根据解码和渲染指令本身的开销确定分块总数的上限;

(2) 根据单机视口范围确定视频水平方向的划分数量和均匀度;

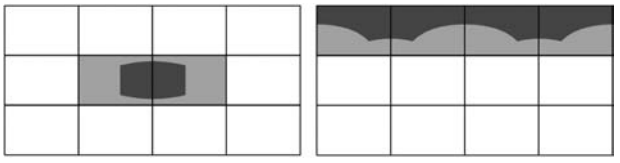
(3) 根据视频内容预测用户的观看兴趣点;

(4) 根据观看兴趣点和屏幕尺寸确定垂直方向的划分数量和均匀度。

整个流程中,步骤(1)与硬件有关,需要通过测试进行估算。集群系统中每个节点的视口范围一般较为接近,因此水平方向可均匀划分。所以实际应用过程中应着重考虑步骤(3)和步骤(4) 确定分块参数。本节将通过两个实例具体说明。

实例 1:全景视频在兴趣点集中于极点附近,如某些标志性高大建筑。由于 ERP 的像素分布是不均匀的,将其映射到球幕空间时,极点附近的像素密度远高

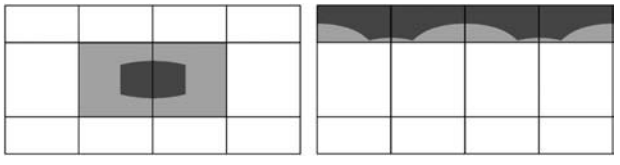
于赤道附近,因此视角旋转到极点附近时会覆盖更多的分块。如图 3 所示,一个 8K(7 680 ×3 840)的全景视频采用 4 ×3 的均匀分块,每个分块大小为 1 920 × 1 280,RGB 格式大约占 7.5 MB 的空间。对于同样大小的 FOV,极点附近覆盖了 4 个视频分块,需处理 7.5 × 4 = 30 MB 的数据,而赤道附近只覆盖了 2 个视频分块,需处理 7.5 ×2 = 15 MB 的数据,因此可能出现赤道附近视频流畅播放,而极点附近视频掉帧的现象。



(a) 视角位于赤道附近 (b) 视角位于极点附近

图 3 均匀分块策略下相同大小的 FOV 在不同视角下覆盖的视频分块

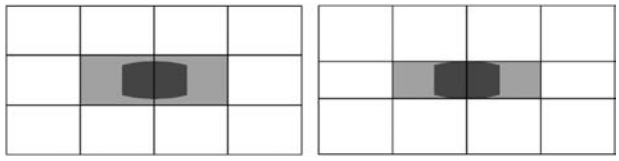
对此我们可以调整竖直方向的分块均匀度,以减少极点附近的解码开销。如图 4 所示,该 8K 的全景视频横向依然均匀分四块,纵向则按照 1∶2∶1 的比例分为三块,因此第 1、3 行的分块大小为 1 920 ×960,RGB 格式大约占 6 MB 的空间,第 2 行的分块大小为 1 920 × 1 920,RGB 格式大约占 12 MB 的空间。此时极点附近覆盖 4 个视频分块,需处理 6 ×4 = 24 MB 的数据,而赤道附近占 2 个分块,需处理 12 ×2 = 24 MB 的数据。可以看到同样是 4 列 3 行的分块,此时极点和赤道附近所需处理的数据量相同。



(a) 视角位于赤道附近 (b) 视角位于极点附近

图 4 非均匀分块策略下相同大小的 FOV 在不同视角下覆盖的视频分块

实例 2:全景视频的兴趣点集中于赤道附近,用户观看时视角主要在水平方向旋转。此时可使用纵向非均匀分块使屏幕显示区域与屏幕所覆盖的分块区域尽可能吻合。图 5(a)中一个 8K 的全景视频采用 4 ×3 的均匀分块策略,每个分块大小为1 920 ×1 280,RGB 格式大约占 7.5 MB 的空间。屏幕显示区域覆盖了 2 个分块,需处理 7.5 ×2 = 15 MB 的数据。而图 5(b)中,该视频按照纵向 1.5∶1∶1.5 的比例分块即可使得赤道区域的分块在纵向上覆盖整个屏幕。此时,赤道附近的分块大小为 1 920 ×960,RGB 格式大约占 6 MB 的空间,需要处理的总数据量为 6 × 2 = 12 MB < 15 MB。



(a) 4 ×3 均匀分块 (b) 4 ×3 非均匀分块

图 5 针对屏幕尺寸的非均匀分块优化

3 分块全景视频转码生成工具

针对上述全景视频分块编码格式,本文实现了相应的视频转码生成工具。对于点播全景视频,我们实现了离线转码工具,预先将其转码成分块全景视频格式的文件;而对于直播视频,我们实现了实时转码组件,将网络直播流实时转码为分块全景视频格式的内存数据流。本文使用 FFmpeg + JPEG - Turbo 对进行视频转码:首先使用 FFmpeg 解码出 RGB 视频帧,然后根据配置将该视频帧划分成多个分块,接下来使用 JPEG-Turbo 对每个视频帧分块进行压缩,最后组成一帧分块编码格式的视频数据。该转码工具的整体框架如图 6 所示。

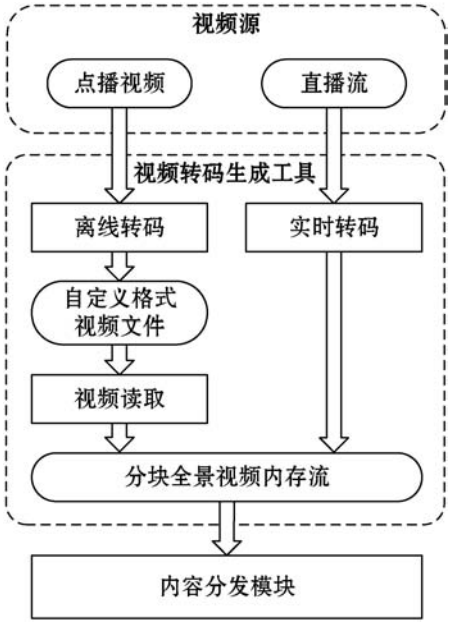


图 6 分块全景视频转码生成工具框架图

离线转码方面,本文实现了一套可视化转码工具,用户输入原视频路径、分块全景视频输出路径、立体格式、分辨率、压缩质量、编号区间以及分块配置等信息,将原视频转码为自定义分块视频文件格式,如图 7 所示。该自定义视频文件由 Header 和 Payload 两部分组成:Header 部分包含视频的基本信息(分辨率、帧率、时长)、分块配置信息以及视频跳转表,其中跳转表的每个表项存储了某一帧号视频数据在整个文件中的偏

移量,视频跳转时检索该表即可;Payload 部分则依次存储了视频每一帧每个分块的数据。

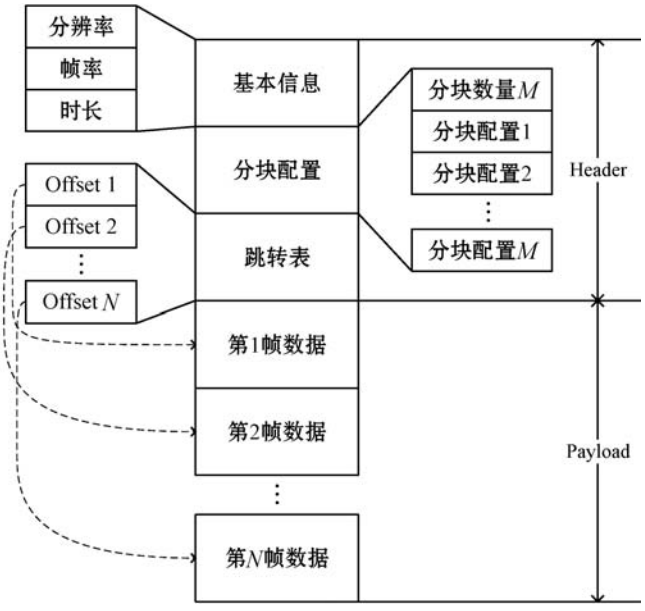


图 7　自定义分块视频文件格式示意图

实时转码组件可将直播视频流根据分块配置实时转码为分块全景视频内存流。实时转码对性能有很高的要求,如视频的帧率是 30 fps,那么平均每帧的转码时间就不能高于 33 ms,因此我们需要对转码效率进行优化。实时转码由直播流解码和分块图像压缩两个步骤组成。直播流解码由 FFmpeg 完成,一般使用解码流水线进行优化,该技术已经较为成熟。分块图像压缩方面,本文使用并发压缩技术优化转码性能。

本文使用线程池进行分块图像的并发压缩。最常用的线程池数据结构基于生产者消费者模式设计,如图 8 所示。线程池维护一个任务队列,存储需要执行的任务列表,并对外提供添加任务的接口。线程池初始化时创建若干线程,每个线程从任务列表中获取任务并执行。这种方案实现简单,逻辑清晰,但是用在超高清视频帧的并发压缩上却存在内存占用过高的问题。本系统中,任务列表中的每个任务存储了未压缩的 RGB 格式数据,由于全景视频的分辨率很高,RGB 格式数据占用的空间很大,如一个 8K × 4K 的视频,每帧数据需要占用大约 100 MB 的空间,如果任务列表容量为 8,CPU 逻辑核心数为 12,那么在转码过程中该线程池最高可能占用(8 + 12) × 100 MB = 2 GB 的内存空间,而整个流媒体服务器的 FFmpeg 解码、网络传输模块都需要使用大量内存。系统运行时,内存空间较少的机器将使用虚拟内存机制将部分内存转移到磁盘,从而导致整个系统性能下降。

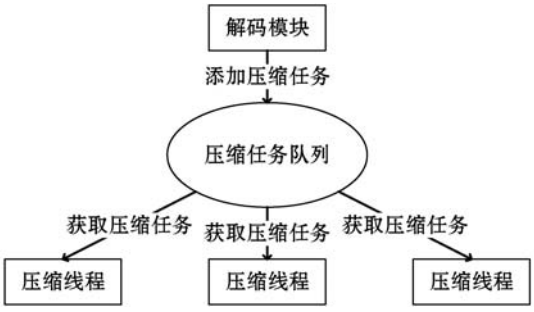


图 8　基于生产者消费者模式的线程池

针对这一问题,本文采用了另一种内存池的设计方式:内存池不再维护一个任务列表,而是直接管理线程,如图 9 所示。线程池维护一个线程队列,初始化时创建与 CPU 逻辑核心数相同的线程到线程队列中。当需要压缩一个视频帧时,从线程队列中取出一个线程,将该视频帧的压缩计算设置为其当前任务;每个线程完成其压缩任务后,将自己重新添加到线程队列中。这种策略虽然设计上较为复杂,但是其内存占用不会超过(12 + 1) × 100 MB = 1 300 MB,相比于前一个方案显著优化了内存使用。

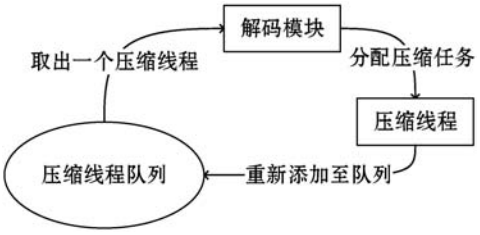


图 9　内存优化的线程池

4　实　验

4.1　实验环境

本文搭建了多投影显示系统对该全景视频分块编码格式进行实验。实验使用的多投影显示系统采用被动立体技术,使用两台计算机分别负责左右眼的画面渲染,操作系统和硬件配置如表 1 所示。每台计算机连接 3 台投影机,使用多投影校正技术拼接融合出完整连续的画面,投影到一个高 2.6 米、半径 4 米的 120° 柱面屏幕上。用户站在该柱面屏幕前,佩戴偏振立体眼镜观看超高清的全景视频。

表 1　多投影显示系统显示节点的操作系统和硬件配置表

计算机节点类型	操作系统 & 硬件配置
多投影显示系统显示节点 1	Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz Windows 7-64 bit,16 GB 内存 NVIDIA Geforce GTX 1060
多投影显示系统显示节点 2	Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz Windows 7 - 64 bit,8 GB 内存 NVIDIA Geforce GTX 970

本文使用 Insta 360 Pro 全景相机搭建全景视频直播流实验环境。该相机支持4K@30 fps 的全景视频直播流拍摄,局域网内可通过 RTMP 协议获取直播流数据。直播流实验使用的网络硬件设备为千兆以太网。

4.2 实验结果与分析

对该分块编码全景视频的整体播放效果进行实验,图 10 展示了该多投影显示系统中使用被动立体技术播放 8K 3D 全景视频的效果图。

图 10 8K 3D 全景视频播放效果图

表 2 给出了该多投影显示系统中不同分辨率分块编码全景视频的播放帧率,所有视频均为 4 ×4 均匀分块。实验结果表明点播视频从 6K 到 12K 均可流畅播放,直播方面则可流畅播放 Insta 360 Pro 相机实时拍摄的 4K@30 fps 全景视频。

表 2 不同分辨率分块全景视频播放帧率统计表

视频信息	点播视频				直播视频
	6K 2D 30 fps	8K 2D 30 fps	10K 2D 30 fps	12K 2D 30 fps	4K 2D 30 fps
播放帧率/fps	30	30	30	30	30

下面对本文分块编码方法的各个技术细节进行实验。首先对分块的随机访问效率进行实验,使用 FFmpeg 进行 HEVC 视频的解码,除了软解码,FFmpeg 还支持 CUVID 硬解码,因此我们比较 HEVC 软解码、HEVC 硬解码和本文 JPEG 图像序列解码三种情况下视频分块随机访问的开销。

测试视频的总分辨率为 7 680 ×3 840@29.97 fps,4 ×4 均匀分块,每个分块大小为 1 920 ×960,HEVC 视频的关键帧间隔为 1 秒。我们在 0 ~4 秒之间随机采样 100 次,记录三种情况下从开始跳转到解码出相应视频帧的时长,实验结果如图 11 所示。可以看到,JPEG 图像序列方案的随机访问解码时长远低于 HEVC 软解码,并且比 HEVC 硬解码要稳定许多。因此 JPEG 图像序列的编码方式可以有效降低随机访问的延迟,缩短分块预测错误时补偿解码的等待时间,从而在不明显影响播放流畅度的前提下,给用户呈现高清晰度的视频画面。

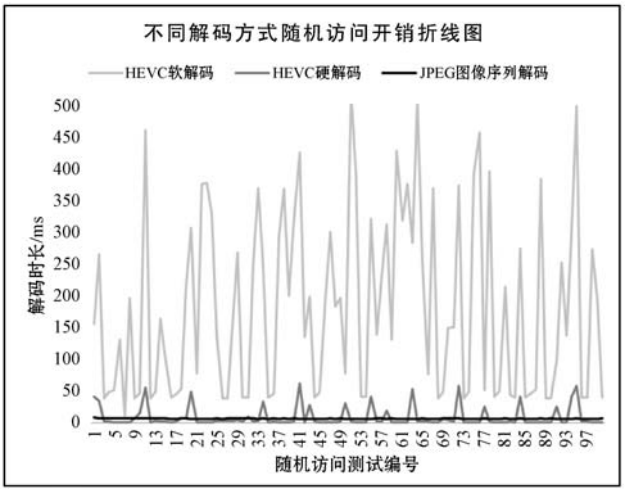


图 11 不同解码方式随机访问开销折线图

分块策略方面,我们测试使用分块编码与否对视频播放性能的影响。使用离线转码工具生成 4K(3 840 × 1 920)、6K(5 760 × 2 880)、8K(7 680 × 3 840)、10K(9 600 ×4 800)、12K(11 520 ×5 760)五种不同分辨率的分块编码全景视频,帧率均为 29.97 fps,每种视频包含未分块和 4 ×4 均匀分块两个副本。虚拟相机的视口大小根据柱面屏幕参数计算得到。图 12 展示了不同视频分辨率下,使用分块编码与否对视频播放帧率的影响。可以看到,视频分辨率不超过 8K 时,不论是否使用分块编码,视频均可按照视频本身的帧率(29.97 fps)播放;而当视频分辨率达到 10K 及 12K 时,非分块的视频播放帧率明显下降,而 4 ×4 均匀分块的视频仍然可以流畅播放。因此分块编码可以显著提高超高清(10K 及以上)全景视频的播放性能。

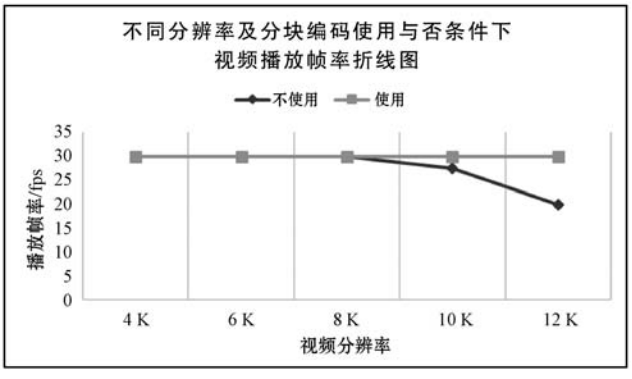
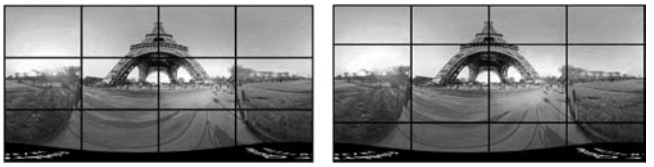


图 12 不同分辨率及分块编码启用与否条件下视频播放帧率折线图

分块均匀性方面,第 2 节讨论了两种非均匀分块带来更佳播放性能的实例,这里分别进行测试。首先测试全景视频在兴趣点集中于极点附近的情况。我们将一部 8K(7 680 ×3 840)@ 29.97 fps 的全景视频转码为均匀 4 ×3 和非均匀 4 ×3 两种分块编码全景视频,其中非均匀分块的纵向比例为 1∶2∶1,如图 13 所示。该视频的兴趣点为埃菲尔铁塔塔尖,主要位于北

极点附近。分块全景视频的分辨率和帧率与原视频相同,虚拟相机的视口大小根据柱面屏幕参数计算得到。



(a) 4×3 均匀分块　　(b) 4×3 非均匀分块

图 13　均匀分块与非均匀分块示意图 1

图 14 展示了两种均匀性配置下,观察视角为赤道附近和极点附近解码 100 帧的总时间,欧拉角分别为(0,0,0)和(180,0,0)。可以看到,非均匀分块在极点附近的解码时间低于均匀分块策略。此外非均匀分块极点和赤道附近的解码时间相差较小,而均匀分块策略则相差较大。因此非均匀分块优化了视角位于极点附近的播放性能,使得极点与赤道附近的解码时间变得均匀。

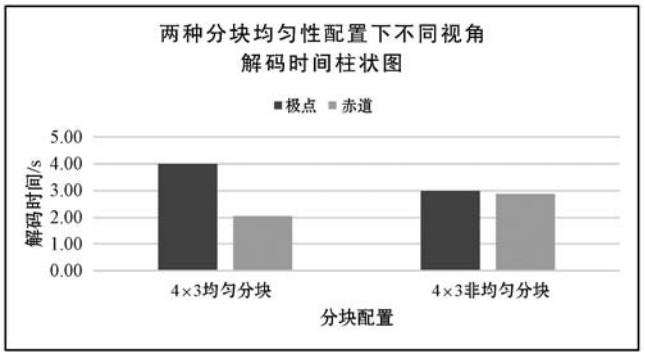


图 14　两种分块均匀性配置下不同视角解码时间柱状图

另一实例中,全景视频的兴趣点集中于赤道附近,用户观看时视角主要在水平方向旋转,此时可根据屏幕尺寸定制分块参数。我们将 8K(7 680×3 840)@29.97 fps 的全景视频转码为均匀 4×3 和非均匀 4×3 两种分块编码全景视频,其中非均匀分块的纵向比例为 1.5∶1∶1.5,如图 15 所示。分块全景视频的分辨率和帧率与原视频相同。视口大小由屏幕尺寸计算得到,视口的朝向为全景视频球幕空间的正中央赤道区域。表 3 展示了屏幕尺寸一定时,这两种分块均匀度配置条件下解码 100 帧的总时间。可以看到,纵向 1.5∶1∶1.5 非均匀分块策略的解码时间低于均匀分块。因此针对屏幕尺寸定制非均匀分块配置可以在不增加分块数量的条件下优化播放性能。

(a) 4×3 均匀分块　　(b) 4×3 非均匀分块

图 15　均匀分块与非均匀分块示意图 2

表 3　屏幕尺寸相同时两种分块均匀性配置下解码时间表

分块配置	4×3 均匀分块	4×3 非均匀分块
解码时间/s	2.03	1.61

最后我们测试第 3 节介绍的并发压缩机制对实时转码性能的影响。我们使用一台 Intel(R) Core(TM) i7-8700K CPU @ 3.70 GHz,6 核 12 线程的机器作为硬件设备。非并发模式下创建单个线程串行压缩,并发模式下则创建 12 个并发的压缩线程。图 16 展示了不同直播流分辨率下使用并发压缩与否对转码时长的影响,纵轴为服务器转码 100 帧视频数据的总时间,其中 3 840×3 840 分辨率的直播流帧率为 24 fps,其余为 30 fps。与解码视频文件不同的是,直播流按照其帧率生成视频帧,因此转码速度不会超过直播流的帧率。可以看到,分辨率较低时并发与串行压缩的效率差别不大,分辨率较高时并发压缩显著提高了实时转码的效率。

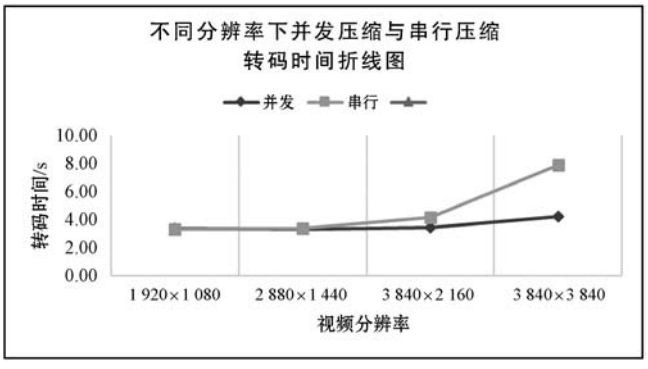


图 16　不同分辨率下并发压缩与串行压缩转码时长折线图

5　结　语

本文针对多投影显示设备的特点,设计并实现了一种面向多投影显示的全景视频分块编码方法,减少单机的性能开销。该方法以已有的 JPEG 分块编码研究[11]为基础,添加对立体全景视频分块编码的支持,实现双分辨率视频流动态适配机制,保证视角变化时视频的流畅播放;添加非均匀分块编码的支持以针对实际需求定制分块参数,优化播放性能。最后,本文使用 FFmpeg 和 JPEG-Turbo 开源库实现了该分块全景视频的转码生成工具,支持离线转码和实时转码两种方式,分别用于点播视频和直播视频。实验结果表明,该分块编码策略可以有效地提升多投影显示设备中高分辨率全景视频播放的流畅度。未来将继续优化该分块全景视频转码、传输、解码和渲染等各个模块的性能,以支持更高分辨率全景视频的流畅播放。

(下转第 165 页)

segments of text[M]//Advances in Information Retrieval. Springer Berlin Heidelberg, 2007:16 - 27.

[4] Yang L, Qiu M, Qu C, et al. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems[C]//The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2018: 245 - 254.

[5] Tur G, Hakkani-Tür D, Heck L. What is left to be understood in ATIS? [C]//Spoken Language Technology Workshop. 2011.

[6] Qu C, Yang L, Croft W B, et al. Analyzing and characterizing user intent in information-seeking conversations[C]// SIGIR'18, July 8 - 12, 2018, Ann Arbor, MI, USA,2018.

[7] Olney A, Louwerse M, Matthews E, et al. Utterance classification in auto tutor[C]//Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing—Volume 2, 2003.

[8] Hutto C J, Gilbert E. VADER: A parsimonious rule-based model for sentiment analysis of social media text[C]//Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media, At Ann Arbor, MI,2014.

[9] Liu B, Lane I. Attention-based recurrent neural network models for joint intent detection and slot filling[EB]. arXiv: 1609.01454, 2016.

[10] Ding X, Liu T, Duan J, et al. Mining user consumption intention from social media using domain adaptive convolutional neural network[C]//Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015:2389 - 2395.

[11] Collobert R, Weston J, Bottou L, et al. Natural language processing(almost) from scratch[J]. Journal of Machine Learning Research, 2011,12(7):2493 - 2537.

[12] Zhang B, Kou Y, Meng W U, et al. Close-range air combat situation assessment using deep belief network[J]. Journal of Beijing University of Aeronautics & Astronautics, 2017, 43(7):1450 - 1459.

[13] Deng L, Hasegawajohnson M, He X. Random features for kernel deep convex network[C]//IEEE International Conference on Acoustics. IEEE, 2013.

[14] Tur G, Li D, Hakkani-Tür D, et al. Towards deeper understanding: Deep convex networks for semantic utterance classification[C]//IEEE International Conference on Acoustics. 2012.

[15] Zhang X, Zhao J, Lecun Y. Character-level convolutional networks for text classification[C]//International Conference on Neural Information Processing Systems. MIT Press,2015: 649 - 657.

[16] Kim Y, Jernite Y, Sontag D, et al. Character-aware neural language models[EB]. arXiv:1508.06615, 2015.

[17] Santos C N, Gatti M A. Deep convolutional neural networks for sentiment analysis of short texts[C]//Proceedings of International Conference on Computational Linguistics. 2014: 69 - 78.

[18] Broder A Z. A taxonomy of web search[J]. ACM SIGIR Forum, 2002, 36(2):3 - 10.

[19] Yao L, Mao C, Luo Y. Graph Convolutional Networks for Text Classification[C]//33rd AAAI Conference on Artificial Intelligence, 2018.

(上接第 155 页)

参 考 文 献

[1] Li J, Wen Z, Li S, et al. Novel tile segmentation scheme for omnidirectional video[C]//2016 IEEE International Conference on Image Processing(ICIP). IEEE, 2016: 370 - 374.

[2] Misra K, Segall A, Horowitz M, et al. An overview of tiles in HEVC[J]. IEEE journal of selected topics in signal processing, 2013, 7(6): 969 - 977.

[3] Skupin R, Sanchez Y, Hellge C, et al. Tile based HEVC video for head mounted displays[C]//2016 IEEE International Symposium on Multimedia(ISM). IEEE, 2016: 399 - 400.

[4] Zare A, Aminlou A, Hannuksela M M, et al. HEVC-compliant tile-based streaming of panoramic video for virtual reality applications[C]//Proceedings of the 24th ACM international conference on Multimedia. ACM, 2016: 601 - 605.

[5] Corbillon X, Simon G, Devlic A, et al. Viewport-adaptive navigable 360-degree video delivery[C]//2017 IEEE international conference on communications(ICC). IEEE, 2017: 1 - 7.

[6] Hosseini M,Swaminathan V. Adaptive 360 VR video streaming based on MPEG-DASH SRD[C]//2016 IEEE International Symposium on Multimedia(ISM). IEEE, 2016: 407 - 408.

[7] Fan C L, Lee J, Lo W C, et al. Fixation prediction for 360 video streaming in head-mounted virtual reality[C]//Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video. ACM, 2017: 67 - 72.

[8] Sullivan G J, Ohm J R, Han W J, et al. Overview of the high efficiency video coding(HEVC) standard[J]. IEEE Transactions on circuits and systems for video technology, 2012, 22(12): 1649 - 1668.

[9] Chen Z, Li Y, Zhang Y. Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation [J]. Signal Processing, 2018, 146: 66 - 78.

[10] 董振江, 张东卓, 黄成, 等. 虚拟现实视频处理与传输技术[J]. 电信科学, 2017, 33(8): 45 - 52.

[11] 杨帆. 面向头盔显示的全景视频播放系统[D]. 上海:复旦大学, 2016.