虚拟环境下的低延迟协同渲染方法研究

王雅妮,高全力,王西汉,毕明洋,焦子逊
(西安工程大学 计算机科学学院,陕西 西安 710600)

摘　要:虚拟现实技术以其独特的沉浸感、交互性,在游戏、直播等各领域的应用愈加广泛。渲染技术是虚拟现实的核心环节之一,而只在本地客户端渲染是目前主要渲染方案,由于实时渲染需要大量的计算资源,导致虚拟现实应用存在响应延迟高、画面刷新率低、图像质量差以及虚拟场景复杂等问题。针对上述问题,该文提出了虚拟环境下低延迟协同渲染方法。首先,将复杂的虚拟环境进行前景交互和背景环境拆分;其次,将渲染工作量较大的背景环境卸载至服务端进行处理;最后,通过运动轨迹预测将渲染的复杂背景以全景图的方式进行分割编码传输至客户端与前景渲染的内容进行合并呈现。经过实验验证,这种协同渲染的方法能够进一步降低传输过程中的 CPU/GPU 利用率以及网络延迟,提高帧刷新率,进而实现高清低延迟交互。

关键词:虚拟现实;人机交互;协同渲染;全景图传输;高清低延迟
中图分类号:TP31　　**文献标识码**:A　　**文章编号**:1673-629X(2023)12-0121-07
doi:10.3969/j.issn.1673-629X.2023.12.017

Research on Low Latency Collaborative Rendering in Virtual Environment

WANG Ya-ni, GAO Quan-li, WANG Xi-han, BI Ming-yang, JIAO Zi-xun
(School of Computer Science, Xi'an Polytechnic University, Xi'an 710600, China)

Abstract: With its unique sense of immersion and interactivity, virtual reality technology is increasingly widely used in various fields such as games and live broadcasting. Rendering technology is one of the core links of virtual reality, and only rendering on the local client is the main rendering scheme at present. Due to the large amount of computing resources required for real-time rendering, virtual reality applications have problems such as high response delay, low image refresh rate, poor image quality and complex virtual scenes. To solve the above problems, we propose a low latency collaborative rendering method in virtual environment. Firstly, the complex virtual environment between foreground interaction and background environment is split. Secondly, the background environment with a high rendering workload is offloaded to the server for processing. Finally, the complex background is segmented and encoded in the way of panorama through motion trajectory prediction and transmitted to the client for merging with the foreground rendered content. After experimental verification, the collaborative rendering method can further reduce the CPU/GPU utilization and network latency in the transmission process, and improve the frame refresh rate, thus achieving high definition and low latency interaction.

Key words: virtual reality; human-computer interaction; collaborative rendering; panorama transmission; high definition and low latency

0　引　言

虚拟现实(Virtual Reality,VR)目前已经在娱乐、医疗保健、教育等众多领域中得到应用。根据市场研究,全球虚拟现实市场预计到 2023 年将达到 51 亿美元,并且在预测期内将超过 55% 的复合年增长率。近年来,VR 线下体验店数量增长迅速,VR 电影、VR 竞技游戏等成为新的体验热点[1-2]。尽管 VR 前景广阔,但仍然面临诸多问题,如虚拟现实技术内容缺乏,渲染过程计算资源庞大,无法实时更新渲染的高清画面,体验过程会出现卡顿眩晕现象,以及用户在体验过程中由于数据线较长存在绊倒的风险等。因此,如何设计出高清、低延迟的交互式虚拟现实系统,在现有的智能终端和无线网络环境下,实现高清、低延迟的交互式 VR 成为当下研究的热点。

收稿日期:2023-01-08　　**修回日期**:2023-05-10
基金项目:国家自然科学基金项目(61902300);陕西省重点产业创新链项目(2020ZDLGY07-05);陕西省教育统计数据研究中心专项研究课题(21JTY010)
作者简介:王雅妮(1996-),女,硕士研究生,研究方向为虚拟现实;通信作者:高全力(1988-),男,博士,副教授,CCF 会员(A7163M),研究方向为大数据感知与处理、边缘计算。

1 相关工作

由于需要渲染高分辨率和高帧率的图形以提供使用过程中的沉浸式体验,VR 应用具有很高的计算负荷。Chang 等人[3]利用维度实验量化了现有 VR 应用的交互时间和定位精度,并研究了目前有线 VR 应用程序以及智能终端 VR 系统的用户体验。Flashback[4]通过设计一种预渲染的方法解决了智能终端的低计算能力问题。Furion[5]系统支持商用移动设备上的高分辨率 VR 应用程序。Xie Chenhao 等人[6]提出了一种新的软硬件相结合的协同设计方案(Q-VR),实现了低延迟、高质量的协同移动 VR。Coterie[7]设计中的关键挑战是确定最佳截止点需要搜索虚拟世界中每个位置的截止点,由于 VR 虚拟世界可能有数亿网格点,这在计算上很难实现。Liu Xing 等人[8]设计并实现了一种新型的移动设备多用户虚拟现实系统—Firefly,Firefly 通过枚举、预渲染、编码和存储在虚拟场景中可到达的所有位置的视图来执行一次性的离线内容准备。

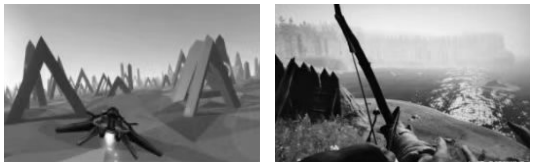
协同渲染的目的是为了营造沉浸感的虚拟现实环境[9]。与上述研究相比,该文针对虚拟场景下渲染资源庞大,渲染环境复杂等问题,提出了客户端服务器协同渲染的方法。该方法将分布式计算技术应用于渲染领域,充分发挥了计算资源的优势,提高了渲染的效率和速度。与上述方法相比较,该方法使用了全景图切片的方式对全景图进行编码压缩传输,一定程度上比视频流传输加载更快,使得用户体验感更好。经实验验证,将上述方法相结合能够降低网络负载,提高帧刷新率,进而实现高清低延迟交互。

2 虚拟场景的前背景区分

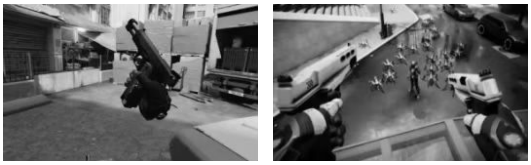
VR 环境渲染需要较好的硬件和网络环境支持,受功耗、散热和体积等物理条件限制,智能终端 CPU/GPU 的运算能力不可能在短时间内得到明显改善[10]。因此将 VR 场景渲染资源进行计算任务迁移,分为前景交互和背景环境交互,虚拟场景下的前景交互和背景环境具有不同的可预测性和渲染工作负载。

2.1 本地渲染的性能瓶颈

对于绝大多数的 VR 应用程序来说,渲染的 VR 内容可拆分为前景交互(FI)与背景环境(BE)。例如,对图 1 中的 VR 应用程序进行了测试,图 1(a)所示的飞行器与控制器相互作用,图 1(b)所示的弓箭,图 1(c)所示的手枪,图 1(d)所示的枪,这些都是前景交互内容,与用户直接相关,这些物体的动画效果会根据用户控制器的输入而改变,虚拟现实技术的基本任务就是构建一个虚拟场景来为用户提供一种沉浸感[11],并以背景环境作为虚拟世界组成的主体,它涵盖了整个游戏空间的绝大部分。



(a)PolyRunner (b)Forest



(c)Alyx (d)Robo Recall

图 1 几款 VR 游戏画面

图 1(b)(c)(d)是高质量的虚拟现实游戏[12]。分别对 3 个游戏的前景和背景环境进行测试,测量出每个游戏版本的帧渲染[13]时间和 CPU,GPU 的利用率。结果如表 1 所示。前景交互时只需要 10 ~ 11 ms,比背景环境的渲染少 8 倍,并且 CPU 和 GPU 的开销都有明显的下降。只渲染背景环境需要 85 ~ 101 ms,并且 CPU 和 GPU 的利用率也是前景交互的 2 ~ 3 倍。由此可见,在 VR 场景渲染过程中,与背景渲染相比较,只渲染前景交互的内容资源利用率较低。其原因就在于高品质 VR 应用下,真实的、沉浸式的环境自然包含了更多的细节和纹理(如图 1 中的建筑和阴影),这导致了巨大的渲染开销。

表 1 3 种高质量 VR 程序资源使用情况

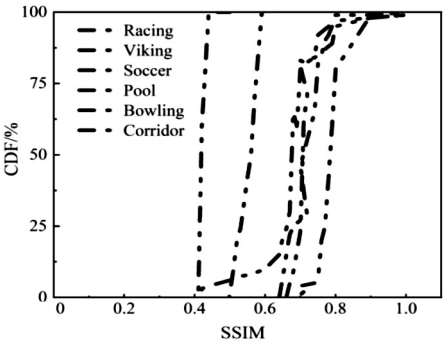
VR 游戏	前景交互 (TPF/CPU/GPU)	背景环境交互 (TPF/CPU/GPU)
Forest	10 ms/28%/20%	85 ms/50%/90%
Alyx	11 ms/35%/22%	93 ms/54%/92%
Robo Recall	10 ms/30%/21%	101 ms/67%/98%

2.2 帧的相似性

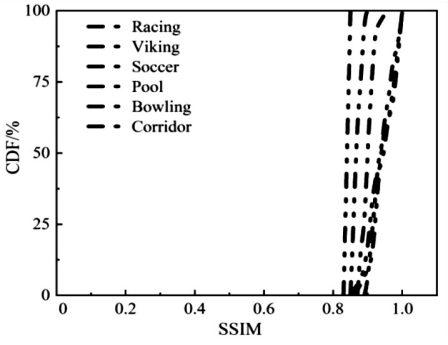
在 VR 应用程序中,影响用户体验跟虚拟场景中玩家的数量、渲染的每帧大小以及帧从服务器传输到客户端频率都有关系[14]。可以通过帧内相似度以及帧间相似度来衡量 VR 程序的渲染方式,以减少背景环境帧的预取频率[15]。

该文测量了用户在虚拟场景中移动时的相邻网格点的背景环境帧之间的相似性,从 Unity Asset Store 的 3D 游戏中分别测量了 6 个 VR 游戏。每个 VR 应用程序分别运行在具有 4K 分辨率的客户端 10 分钟并记录下用户在虚拟世界中的轨迹,然后为轨迹中的每个网格点生成全景的背景环境帧,并计算相邻背景环境帧之间的相似度。图 2 绘制了每个 BE 帧与其在轨迹中的下一个相邻 BE 帧的相似性(CDF)。从图 2(a)中可以看出,在 6 个 VR 应用程序中,前背景未分离时

相邻帧的相似性接近于 0,SSIM 值大于 0.9 的只占不到 20%。因此,在整个程序渲染过程中,每一帧对应相似度并不高。为了衡量虚拟场景中 FI 帧与 BE 帧的分离对帧相似性的影响,使用自适应截断半径的方案生成截断半径,对 6 个 VR 应用程序进行度量,如图 2(b)所示,单个玩家相邻的 BE 帧之间的相似性显著提高。



(a)分离前



(b)分离后

图 2　VR 应用程序在分离前后相邻 BE 帧的相似度

2.3　前背景区分

上述关于玩家附近的物体降低虚拟世界中渲染的 BE 帧的相似性以及在 2.1 的分析中,GPU 在渲染背景环境时资源消耗更多。在程序上,根据截断半径来区分 FI 和 BE,如图 3 所示,其中半径内的建筑物属于前景交互内容,半径外的楼房属于远的背景虚拟环境。

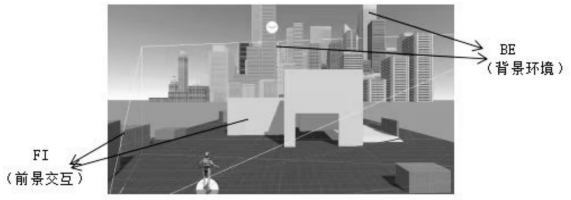


图 3　虚拟场景中前景和背景拆分

由于 BE 往往比 FI 包含更多的交互对象,特别是对于复杂场景的 VR 应用,渲染 BE 在移动设备上仍然具有很大的挑战性,因此,应该在远程服务器上进行 BE 的渲染和预取。FI 包含的对象比 BE 少很多,因此,可以在移动客户端进行渲染。然而,由于 FI 接近于用户的视点,FI 将占据最终帧的重要部分,因此,渲染过程中接近 BE 的帧将保持高质量较大帧。FI 帧和 BE 帧大约是原始 BE 帧的一半。因此,在服务器上渲染 FI 帧不会大大降低网络负载,FI 帧应该在移动客户端进行渲染。

2.4　自适应截断方案

由于 VR 虚拟世界中的物体密度变化大,用户在使用过程中会随时改变移动位置,为每一个位置确定一个截断半径在计算上来说是不可行的,因为 VR 游戏在虚拟世界中可以包含数亿个网格点[16]。因此,该文提出一种自适应截断方案,该方案大大减少了需要计算的截断半径的数量,并且可以递归地划分虚拟世界,直到每个子区域不同位置的截断半径变得大致均匀。流程如图 4 所示。在大多数 VR 环境中,用户是以二维方式在移动,因此虚拟世界的递归分区为二维。对于每个 K 值,遵循上述虚拟世界递归划分叶区域为四叉树,然后测量 FI 中不满足最低延迟条件的区域百分比小于 0.25%,因此在设计中 K 值为 10 是最佳选择。

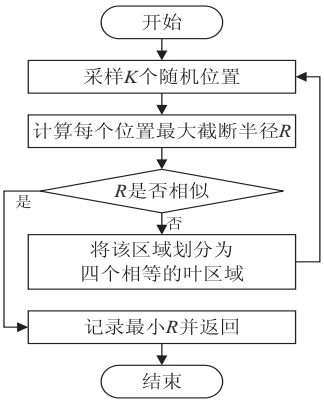


图 4　自适应截断半径流程

该自适应截断方案根据虚拟世界中变化的对象密度自适应地生成截断半径,最小化虚拟世界中不同位置分离前景和背景的截断半径的总数,同时最大化每个叶区域的截断半径,从而最大化其中的位置。

3　协同渲染方法研究

3.1　协同渲染机制

针对 VR 应用程序的渲染负载和 CPU,GPU 利用率等指标进行了实测与分析。该文提出可以利用客户端和服务器的协同渲染机制对虚拟环境的前景交互和背景环境分别进行渲染,主要有以下几个步骤:

(1)在 VR 应用场景中,将总体的 VR 渲染负载拆分成渲染前景交互和渲染背景交互两部分。

(2)前景交互的部分利用本地 GPU 进行渲染。

(3)背景环境的部分利用服务器端进行预加载和预渲染。

(4)最终在客户端将前景交互和背景环境的渲染

效果进行合并,呈现给用户。

使用客户端和服务器协同渲染可以通过计算迁移,很大一部分计算开销被无线网络传送到服务器上实现。本地渲染是随机的并且难以预测,可以避免实时交互受到较大网络延迟,并且可以使用预渲染和预加载的方法,避免网络传输时产生的附加延迟。所以,采用协同渲染的方式,若背景环境能在本地客户端被及时加载,每渲染一帧延迟可减少为本地渲染前景与本地解码背景所需时间的最大值加上合成前后景的时间。协同渲染需要在终端上将前景交互和背景环境合并,实际渲染系统中(例如 OpenGL ES),需要将像素数据填充在一个缓冲区中,进而生成最终画面呈现给用户。

3.2 预加载背景环境帧

在一般的协同渲染器中,如果直接在 VR 应用中抓取下一帧可能会出现诸多问题,由于用户可以随机改变移动方向和位置,并将服务器预渲染的高质量帧传送给客户端,这会带来巨大的带宽开销,同时也会增加客户端 CPU 的利用率,因此,可以通过以下 VR 方案来优化预加载背景环境帧。

客户端请求背景环境帧的指令首先会被发送到帧缓存区,如果缓存区未找到类似帧则会被发送至服务器。在背景环境中,相邻的 BE 帧具有很高的相似性,由于虚拟环境可以离散成数个网格点,则网格点 i 的预取 BE 通常可以重复用于相邻几个网格点。如图 5 所示,网格点 0 处的 BE 帧位于帧缓存区,且 0 处的帧缓存可以重用于附近区域,当用户从 0 位置移动到 2 位置时,只需要在到达 4 位置之前预渲染出 4 以及 5,6,7 位置即可。这种帧的重用方式一方面降低了客户端从服务器预取帧的频率,另一方面每次预取可以留有更大的时间窗口使得 BE 帧能够预加载出来。

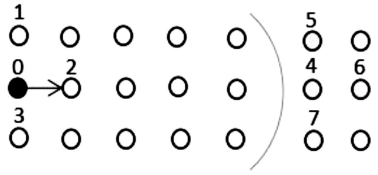


图 5 背景环境帧预取示意图

3.3 帧缓存

虚拟世界中的临近位置可以重用帧缓存,由于上述自适应截断方案可以导致不同叶区域的不同截断半径,因此高速缓存查找算法还需要考虑临近点的其他因素。

帧缓存查找算法:缓存存储的是背景环境的相关数据,比如对应的网格点和所属的叶区域。如果网格点 K 满足以下 3 个条件,则对于网格点 K 的缓存查找所返回的帧缓存结果为找到:(1)网格点在网格点 K 一定的阈值内;(2)由于不同叶区域有不同的截断半径,则其与网格点需要位于同一叶区域;(3)为了确保渲染的 FI 与缓存的 BE 帧合并后没有缺失,其对应的 FI 需包含与网格点 K 相同的对象。这 3 个条件确保重复使用的 BE 与 FI 更好的融合呈现。在满足上述约束所有缓存帧中,最接近正在查找的网格点的帧将作为最相似的帧返回。

3.4 利用全景图切片模型封装节点

在 VR 系统中,控制器将输入用户的姿态,其中包括位置及角度信息,但是,在任意一个网格点,用户可以根据自己的需求任意变换位置和视角,因为看到的内容会随着转动角度的不同而发生变化。所以,很难预测出用户在将来旋转的角度信息。该文提出利用全景图来进行背景环境采集,全景图像覆盖了当前位置 360 度的所有信息,对任何位置来说,不管是哪一种视角下的内容,均可对全景图进行裁剪,获得对应内容。通过全景图切片编码方法,当运动到某个位置时,预先加载周围位置上对应的全景图切片。

全景图的显示过程需要较高的分辨率和较大的数据量,而直接加载在三维场景中会造成系统卡顿。为了获取高效率的渲染运算,该文使用了动态连续多分辨率(LOD)技术[17],在全景图数据中构造图像金字塔模型并依据展示范围对数据进行动态调度。

表 2 全景图金字塔层级与分块数目

层级模型	数据块数目/个	分辨率
0	1	256×256
1	4	512×512
2	16	1 024×1 024
3	64	2 048×2 048
4	356	4 096×4 096
5	1 024	8 192×8 192

图像金字塔是根据 LOD 技术构建的一种多分辨率层级模型——对图像进行分级分块。在虚拟场景下,该文利用全景相机进行全景图像获取,采用 8 192×8 192 高分辨率全景图为原始数据对模型底层进行处理,在向上构建层级模型的过程中,原始数据被重新采样,分辨率层层递减,最上层图像分辨率最小。鉴于所收集全景图的长宽比是 1∶1,适用于四叉树分块方法对数据分块。设定起始数据块的尺寸为 256×256,8 192×8 192分辨率全景图数据分块,图像分辨率如表 2 所示。图像分辨率从低到高被命名为 LOD_i($i=0,1,\cdots,5$)。基于四叉树,以行列编码方式传输数据图像,设左下角数据块为开始数据,坐标为(0,0)。每块图像编码为 xyz,x 表示数据块所在行,y 表示数据块所在列,z 表示数据块所属层级。假设初始在 001 位

置,此时会预加载好 101,111,011 对应的全景图,已被预加载过的全景图将暂存于本地缓存,避免重复预加载,进而减小了预加载延迟,降低了网络传输开销。

3.5　利用编码技术对全景图进行压缩

使用全景图切片模型将会为虚拟世界的每个位置渲染一张全景图,而非对每一个方向渲染一帧图像,这已在很大程度上减少了所需帧数。H.264 编码过程中,每个帧被分成大小一致的宏块。对这些宏块进行离散余弦变换(DCT)、量化和熵编码。8 * 8 模块的快速 DCT 及其反变换的定义如公式(1)(2)。

$$X_{uv} = \frac{1}{4} M_u M_v \left[\sum_{x=0}^{7} \sum_{y=0}^{7} F_{xy} * \cos \frac{(2x+1)u\pi}{16} * \cos \frac{(2y+1)v\pi}{16} \right] \tag{1}$$

$$F_{uv} = \frac{1}{4} M_u M_v \left[\sum_{x=0}^{7} \sum_{y=0}^{7} X_{xy} * \cos \frac{(2x+1)u\pi}{16} * \cos \frac{(2y+1)v\pi}{16} \right] \tag{2}$$

其中,

$$M_u M_v = \begin{cases} \frac{1}{\sqrt{2}}, & \text{当 } u,v \text{ 为 0 时} \\ 1, & \text{其他} \end{cases} \tag{3}$$

其中,F_{uv} 表示未变换的图像数据,X_{uv} 表示通过 DCT 后得到的图像数据。

假定处于虚拟世界的任何地方,存在 n 个潜在的浏览角度,每个普通帧的代销为 10 MB,则下一时刻,用户移动位置可能是自己的位置以及前后左右 4 个位置,需要预加载的普通帧大小为 5 * n * 10 MB。而用整幅全景图加载时,每个全景图像为 4 MB,那么在下一时刻抵达的地点可能用到的全景图大小为 3 * 4 MB,因为每一次抵达的新地点最多为周围 3 个位置。因此,利用相邻全景图的相似的特点,对每个需要加载的全景图编码为 P 帧进行压缩,最终每次只需要加载 130 KB 的全景图。

将全景图加载到客户端后进行解码,由于全景图进行了分片处理,当分片全景图加载到本地后由客户端的解码模块进行解码后与前景交互的内容进行合并,VR 系统的传感和失真校正是通过 HTC Vive SDK 实现的,其传感器需要从 VR 头盔中录入用户信息,畸变校正模块负责把融合的 VR 内容变换成与人眼距离相适应的图像传送到显示器上。

3.6　预加载轨迹预测模型

在 VR 系统中,利用渲染的全景图可以提供给用户更加流畅直观的感受,在移动到某一个位置的时候,会对周围对应的全景图进行预加载,这就需要实时掌握用户的运动轨迹。该文使用了卡尔曼滤波算法对用户的轨迹进行动态跟踪。

卡尔曼滤波(Kalman Filtering)动态轨迹预测方程和观测方程如公式(4)(5):

$$X(k+1) = A(k)X(k) + T(k)W(k) \tag{4}$$

$$Z(k) = H(k)X(k) + V(k) \tag{5}$$

对于预测轨迹点与实际轨迹点的几何空间误差采用公式(6)所示的均方根误差(RMSE)来计算:

$$\mathrm{RMSE} = \frac{\sum_{i=1}^{k} \sqrt{(x_i' - x_i)^2 + (y_i' - y_i)^2}}{k} \tag{6}$$

其中,(x_i, y_i) 代表实际轨迹点的位置信息,(x_i', y_i') 代表预测轨迹点的位置信息,k 为预测轨迹点的数量。当轨迹点预测完成后,根据 RMSE 与给定阈值的大小关系,对轨迹预测结果进行精度判断,在 RMSE 小于阈值时为命中;否则,未命中。

4　实验设计

4.1　实验平台

实验在 PC 机上完成,PC 设置为 Windows 10 操作系统,处理器为 Intel(R) Xeon(R) W-2102 CPU @ 2.90 GHz,机带 RAM 为 8.00 GB,GPU 为 NVIDIA Quadro P400,GPU 内存为 5.8G,unity 版本为 2020.3.12f1c1(64-bit)。

4.2　实验过程

将协同渲染方法服务端运行在一台性能较强的图形工作站上。将客户端部分运行在一台可移动 PC 机上。通过 802.11ac WiFi 网络和服务端相连,此网络能够提供 400 Mbps 的可用带宽,文中方法使用自建 VR 虚拟环境进行验证。

针对虚拟场景设置了 3 种测试版本:(1)仅在本地客户端渲染:VR 内容由本地计算机的 CPU/GPU 渲染;(2)仅在远程服务器渲染:通过 WiFi 网络将所有的渲染任务都迁移至一台高性能图形工作站进行渲染;(3)通过文中方法:通过本地与服务器计算资源协同渲染 VR 内容最终呈现给用户。

4.3　实验结果与分析

该文使用 SSIM 指标来量化系统渲染的图像质量。SSIM 通常被用于评价最终图像质量与原始图像质量之间的质量损失,它的值介于 0 和 1 之间,如果 SSIM 值越接近 1,意味着相似度越高,反之相似度越低。表 3 展示了 VR 应用在 3 种渲染情况下的图像质量以及画面刷新率。在本地客户端渲染情况下,SSIM 值约为 0.8,显示画质相对不高。该结果主要由于本地渲染存在计算开销大,图像渲染不及时所致。远程渲染和协同渲染的 SSIM 均高于 0.9,这是由于在服务器端渲染,传输内容均通过 H.264 进行了压缩和解压缩,其传输的画质内容较高。

表 3 不同渲染方式下的图像质量和刷新率

实现方法	图像质量（平均 SSIM）	刷新率（平均 FPS）
客户端渲染	0.825	30
服务器渲染	0.913	45
协同渲染	0.945	87

为了保证 VR 系统逼真的用户体验，要求系统提供更高的帧数，以实现虚拟场景下更为流畅的画面。依照参考文献[18]，VR 至少要达到 60 FPS 才能使画质流畅。而协同渲染的刷新率平均可以达到 87 FPS。接下来，修改了虚拟场景下的动态对象的数量，同时测量了不同渲染条件下的 FPS 指标。图 6 展示了不同渲染方式下，渲染动态交互对象的数量以及 FPS 的变换。可以看出协同渲染方式下渲染动态交互随着交互数量的增加其刷新率较高且保持稳定。

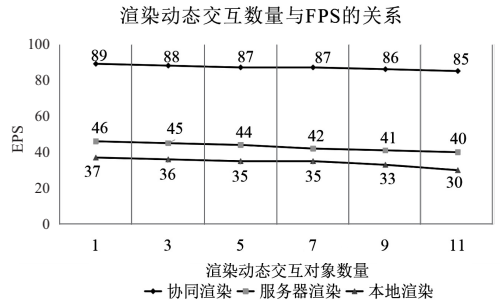


图 6 动态渲染对象数量变化与 FPS 的变化情况

在交互响应性方面，该文测量了在 3 种渲染方式下的 CPU 处理 1 帧所消耗的总时间。由图 7 可以看出，协同渲染下 CPU 处理 1 帧所消耗的总时间平均为 0.9 ms，相对于其他两种方式耗时较短。接下来测量了用户在虚拟场景中转动头显设备时的转动延迟和运动过程中的位移延迟。测量结果表明，相比于其他两种渲染方式，协同渲染可以有效地降低用户操作过程中的响应延迟，转动延迟要低于运动延迟是因为在转动的过程中，任何一个角度的画面都可以由当前位置的全景图切片按需加载而成。

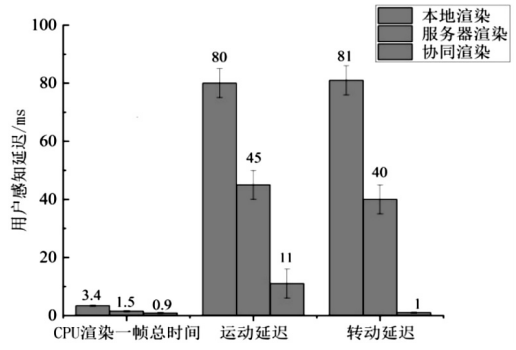


图 7 VR 应用在移动过程中的用户感知延迟

该文对 3 种渲染方式下的 CPU/GPU 资源利用率和运行过程中的网络带宽使用情况进行了实测。从表 4 可以看出，协同渲染方式下 CPU 和 GPU 的利用率以及平均带宽都较低，这些数据表明，将本地较大的计算资源迁移到服务器端进行计算可以有效地提高渲染效率。

表 4 3 种渲染方式下的资源开销情况

渲染方式	CPU 利用率/%	GPU 利用率/%	平均带宽开销/Mbps
本地渲染	61	40	132
服务器渲染	55	43	91
协同渲染	45	24	90

以上实验结果均验证了 VR 应用在协同渲染方式下的性能指标，并且可以在现有的设备条件和无线网络环境下实现高清、低延迟交互。

5 结束语

随着虚拟现实技术的发展，高清低延迟交互必将成为研究的重要课题。该文针对虚拟场景下用户浏览时产生的响应延迟高、画面刷新率低、图像质量差等问题，提出将虚拟场景中复杂的渲染环境进行前景和背景区分，将前景较小的渲染资源放在本地进行计算，背景较大的渲染工作量迁移至另外一台运算较强的服务器端进行协同渲染。实验表明，提出的客户端和服务器协同渲染的方法在刷新率上有了明显提升，并且资源利用率和用户感知延迟也有所降低，此方法可以有效减轻用户在移动过程中的画面卡顿不流畅等问题，进而实现复杂虚拟环境下的高清、低延迟交互。

由于该文是基于现今无线网络进行数据传输，所能达到的最高网络带宽在传输过程中也会造成一定的延迟，随着 5G 技术的发展以及 6G 技术的到来，可以从网络带宽方面进一步提升，从而降低网络传输速率。

参考文献：

[1] 余诗曼，许奕玲，麦筹璋，等. 虚拟现实技术的应用现状及发展研究[J]. 大众标准化，2021(21)：35-37.

[2] SLIVAR I，VLAHOVIC S，SILIC M，et al. The impact of network and social context on quality of experience for competitive multiplayer virtual reality games[C]//Proceedings of the 2nd workshop on games systems. NY：ACM，2022：16-21.

[3] CHANG C M，HSU C H，HSU C F，et al. Performance measurements of virtual reality systems：quantifying the timing and positioning accuracy[C]//ACM on multimedia conference. Amsterdam：ACM，2016：655-659.

[4] BOOS K，CHU D，CUERVO E. Flashback：immersive virtual on mobile devices via rendering memoization[C]//Proceedings of the 14th annual international conference on mobile systems，applications，and services. Singapore：ACM，2016：

291-304.

[5] LAI Z, HU Y C, CUI Y, et al. Furion: engineering high-quality immersive virtual reality on today´s mobile devices [C]//Proceedings of the 23rd annual international conference on mobile computing and networking. Snowbird: ACM, 2017: 409-421.

[6] XIE C, LI X, HU Y, et al. Q-VR: system-level design for future mobile collaborative virtual reality [C]//Proceedings of the 26th ACM international conference on architectural support for programming languages and operating systems. Virtual: ACM, 2021: 587-599.

[7] MENG J, PAUL S, HU Y C. Coterie: exploiting frame similarity to enable high-quality multiplayer vr on commodity mobile devices [C]//Proceedings of the twenty-fifth international conference on architectural support for programming languages and operating systems. Lausanne: ACM, 2020: 923-937.

[8] LIU X, VLACHOU C, QIAN F, et al. Firefly: untethered multi-user VR for commodity mobile devices [C]//Proceedings of the 2020 USENIX conference on USENIX annual technical conference. [s. l.]: USENIX Association, 2020.

[9] ELVEZIO C, LING F, LIU J S, et al. Collaborative virtual reality for low-latency interaction [C]//The 31st annual ACM symposium on user interface software and technology adjunct proceedings. Berlin: ACM, 2018: 179-181.

[10] ABRASH M. What VR could, should, and almost certainly will be within two years [J]. Steam Dev Days Presentation, 2014, 9(1): 42-55.

[11] MARKS B, THOMAS J. Adoption of virtual reality technology in higher education: an evaluation of five teaching semesters in a purpose-designed laboratory [J]. Education and Information Technologies, 2022, 27(1): 1287-1305.

[12] WANG J, SHI R, XIAO Z, et al. Effect of render resolution on gameplay experience, performance, and simulator sickness in virtual reality games [C]//Proceedings of the ACM on computer graphics and interactive techniques. [s. l.]: ACM, 2022: 1-15.

[13] WENG C Y, CURLESS B, SRINIVASAN P P, et al. Human-NeRF: free-viewpoint rendering of moving people from monocular video [C]//2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR). New Orleans: IEEE, 2022: 16189-16199.

[14] RHEE T, THOMPSON S, MEDEIROS D, et al. Augmented virtual teleportation for high-fidelity telecollaboration [J]. IEEE Transactions on Visualization and Computer Graphics, 2020, 26(5): 1923-1933.

[15] ELBAMBY M S, PERFECTO C, BENNIS M, et al. Toward low-latency and ultra-reliable virtual reality [J]. IEEE Network, 2018, 32(2): 78-84.

[16] DANG P, ZHU J, PIRASTEH S, et al. A chain navigation grid based on cellular automata for large-scale crowd evacuation in virtual reality [J]. International Journal of Applied Earth Observation and Geoinformation, 2021, 103: 102507.

[17] 王振武,吕小华,韩晓辉. 基于四叉树分割的地形 LOD 技术综述[J]. 计算机科学,2018,45(4):34-45.

[18] TALEB T, NADIR Z, FLINCK H, et al. Extremely interactive and low-latency services in 5G and beyond mobile systems [J]. IEEE Communications Standards Magazine, 2021, 5(2): 114-119.