Journal of System Simulation

Volume 31 | Issue 8 Article 13

12-12-2019

GI-Map Tree: Global Illumination Collaborative Real-Time Rendering in Web3D Dynamic Scene

Liu Chang
1. Tongji University, Shanghai 201804, China; ;2. Nanchang Aviation University, Nanchang 330063, China; ;

Jinyuan Jia
1. Tongji University, Shanghai 201804, China; ;

Yifan Lu
2. Nanchang Aviation University, Nanchang 330063, China; ;

Zhang Qian
3. Shinegraph Information and Technology Co., Ltd, Shanghai 201804, China; ;

See next page for additional authors

GI-Map Tree: Global Illumination Collaborative Real-Time Rendering in Web3D Dynamic Scene

Abstract

Abstract: For the dynamic illumination in web-based virtual reality environment, *an interactive real-time rendering mechanism that integrates pre-storage, scheduling and optimization based on a collaborative real-time rendering system*. The mechanism helps the rendering system to construct a viewpoint-based pre-storage structure, “GI map tree”, to save and organize illumination rendering information on cloud server. With the assist of “GI map tree” and the user's input behavior, a “heat”-based GI map group scheduling algorithm is proposed to improve the transmission efficiency of the rendering system in this mechanism. The experiment results of Web3D application in the Web environment show that the mechanism can not only effectively organize thousands of GI maps, but also help the collaborative real-time rendering system to accelerate the reconstruction of dynamic scene illumination.

Keywords

collaborative real-time rendering, dynamic illumination, web-based virtual reality, GI map tree, 3D image warping

Authors

Liu Chang, Jinyuan Jia, Yifan Lu, Zhang Qian, and Zhao Lei

Recommended Citation

Liu Chang, Jia Jinyuan, Lu Yifan, Zhang Qian, Zhao Lei. GI-Map Tree: Global Illumination Collaborative Real-Time Rendering in Web3D Dynamic Scene[J]. Journal of System Simulation, 2019, 31(8): 1591-1604.

第 31 卷第 8 期 系统仿真学报© Vol. 31 No. 8
2019 年 8 月 Journal of System Simulation Aug., 2019

光图树：Web3D 动态场景全局光照的协同实时渲染

刘畅 [1,2]，贾金原 [1]，陆一凡 [2]，张乾 [3]，赵磊 [4]
（1. 同济大学，上海 201804；2. 南昌航空大学，南昌 330063；3.上海渲图信息科技有限公司，上海 201804；4.电子科技大学，成都 610054）

摘要：针对于网页虚拟现实环境下的动态光照，在协同式实时渲染系统的基础上*提出了一套集预存储、调度及优化为一体的协同实时渲染机制*。该机制协助渲染系统在云端构建了基于视点信息的八叉树预存储结构“光图树”以保存和组织光照渲染信息。借助“光图树”以及用户的输入行为，机制引入了基于热度信息的光图组调度算法以提升渲染系统的传输效率。Web3D 应用的测试结果表明本机制不但能够高效梳理数以千记的全局光照贴图，而且可以协助协同式实时渲染系统对动态场景光照加速重建。

关键词：协同式实时渲染；动态光照；网页虚拟现实；光图树；三维图像变换
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2019) 08-1591-14
DOI: 10.16182/j.issn1004731x.joss.18-VR0731

GI-Map Tree: Global Illumination Collaborative Real-Time Rendering in Web3D Dynamic Scene

Liu Chang[1,2], *Jia Jinyuan*[1], *Lu Yifan*[2], *Zhang Qian*[3], *Zhao Lei*[4]
(1. Tongji University, Shanghai 201804, China ; 2. Nanchang Aviation University, Nanchang 330063, China; 3. Shinegraph Information and Technology Co., Ltd, Shanghai 201804, China; 4. University of Electronic Science and Technology, Chengdu 610054, China)

Abstract: For the dynamic illumination in web-based virtual reality environment, *an interactive real-time rendering mechanism that integrates pre-storage, scheduling and optimization based on a collaborative real-time rendering system*. The mechanism helps the rendering system to construct a viewpoint-based pre-storage structure, “GI map tree”, to save and organize illumination rendering information on cloud server. With the assist of “GI map tree” and the user's input behavior, a “heat”-based GI map group scheduling algorithm is proposed to improve the transmission efficiency of the rendering system in this mechanism. The experiment results of Web3D application in the Web environment show that the mechanism can not only effectively organize thousands of GI maps, but also help the collaborative real-time rendering system to accelerate the reconstruction of dynamic scene illumination.

Keywords: collaborative real-time rendering; dynamic illumination; web-based virtual reality; GI map tree; 3D image warping

引言

近年来，随着网页三维(Web3D)技术不断提升，基于网页(Web)虚拟环境的 3D 应用在各个领域都层出不穷，如：教育与培训、视频游戏、虚拟现实等。作为 Web3D 的当前最为流行的核心 API—WebGL，已经被绝大多数的浏览器所支持。紧随而来的各种 WebGL 功能库的出现，如：Three.js，Babylon.js 和 Blend4Web 等，降低了开发者在 Web3D 应用项目中的开发难度。

实际上，Web3D 应用都要使用到 Web 虚拟环

收稿日期：2018-07-31 修回日期：2018-10-31；
基金项目：同济大学中央高校基本科研业务费专项资金(2100219066)，中央高校基本科研业务费(0200219153)，江西省教育厅科技项目(GJJ160697)；
作者简介：刘畅(1983-)，男，江西南昌，博士，研究方向为虚拟现实、计算机图形学。

境下的 3D 实时渲染技术，同时每个 Web3D 应用都有着各自的 3D 实时渲染系统。根据渲染任务的分配策略，本文将所有在 Web 虚拟环境下的实时渲染系统分成 3 类：

1) 本地实时渲染系统，该系统把主要的渲染任务放在 Web 前端。它是目前大部分 Web3D 应用的主流渲染系统，有着跨平台性好、易部署等优点。然而由于 Web3D 渲染技术的核心 API—WebGL，AGAL 等渲染能力的局限性，该系统在具有高渲染质量要求的 Web3D 应用中表现不足。

2) 远程实时渲染系统，此系统把主要的渲染任务放在服务器端，并以图片流(图片帧序列)的形式把渲染的结果传递到 Web 前端。此时的 Web 前端仅被用来显示渲染结果而不参与渲染任务。该系统具有优秀的渲染质量，然而传输延时所导致的缺帧现象是该系统无法避免的缺陷。

3) 协同式实时渲染系统，此系统的渲染任务选择性的在 Web 浏览器和服务器端同时进行，渲染开销较低的任务通常放在 Web 前端，反之则放在服务器端。最终，两端的渲染结果将同时在 Web 前端混合输出。该系统充分发挥了服务器端的渲染能力强的优势，同时也没有浪费 Web 前端的渲染资源。然而交互性传输延时带来前后端渲染结果差异是困扰该系统的主要问题。

本文选择协同式协同式渲染系统——Cloud Baking[1-2]作为 Web 虚拟环境下实时渲染系统主要的解决方案。在 Cloud Baking 中，云后端进行复杂且高计算强度的光影渲染，如：全局光照渲染、软阴影等，并将渲染结果保存到光照图片(简称光图)中；Web 前端只进行轻度的直接光照渲染，所得渲染帧再与接收自云后端的光图进行混合以得到在 Web 浏览器中演示的最终效果帧。该解决方案在分别继承了前两种系统中跨平台、易部署和渲染质量佳等优点的同时又规避了 Web 前端渲染能力弱的缺陷。

然而，Cloud Baking 需要在云服务端实时渲染大量的光图样本以显示具有高视觉质量的复杂虚拟场景，尤其在动态虚拟环境中。这使得 Cloud Baking 成为了一个渲染资源消耗大，网络带宽要求高的计算密集型系统。因此，本文提出光图树渲染机制以减少云服务器的渲染时长、降低传输带宽需求并提升输出结果的渲染质量。主要贡献如下：

1) 提出光图树存储结构，结构保存并梳理云服务器端已渲染的光图。这不仅极大提升了光图的使用率，而且使得的云服务器避免重复渲染相同视角下未发生光照变换的光图。

2) 提出的热度优先光图搜索算法，算法以用户行为区域为考量依据，将行为密集区域作为优先搜索范围，以此提高了光图搜索率。

3) 提出基于纹理映射的三维图像变换法“3D texture warping”，该方法在协同式渲染系统的特定环境下，将传统三维图像变换法(3D image warping)中的像素偏移转换为纹理映射，不仅在算法复杂度上有极大的改善而且提升了交互延时问题的优化效率。

1 相关工作

1.1 协同式实时渲染

协同式实时渲染充分利用了前后端的渲染资源，使得渲染效率达到最佳。在虚拟现实环境下，即使有强大的渲染后端，但客户前端的本地渲染仍然可以协助诸多渲染系统减少各自交互延时，这些系统包括：虚拟漫游系统[3]，远程视觉系统[4]和云游戏系统[5]。协同式实时渲染系统的关键是保持本地渲染与服务器渲染的同步。对于可视化或漫游应用程序，服务器与客户端同步只需要保持渲染视点的一致性，而对于游戏应用则由于有自身游戏逻辑原因变得更为复杂。Dead Reckoning[6]是一种在开发多玩家在线游戏时广泛采用的逻辑同步策略，可用于解决这一问题。

1.2 基于图像的渲染

基于图像的渲染(Image Based Rendering，IBR)技术是以图像作为主要输入，并输出全新视图的渲染技术，在本文中光图的存储、组织、调度和传输都与之紧密相连。该技术依据几何信息参与程度被分为 3 类[7]，分别是：

1) 不带几何信息，此类 IBR 技术以大量图像样本为输入，并以 plenoptic 函数[8]为基础。比较典型研究包括：光场渲染[9]，Lumigraph[10]，mosaics[11]和 panoramas[12]。其特点是输入数据庞大，传输效率低下，不利于实时渲染。

2) 带隐式几何信息，此类技术借助少量图像的拍摄相机参数来渲染新的视图，参与渲染计算的 3D 几何信息通常都是通过图像反投影而来。以视点重建为目标的视图插值[13]和视图变形[14]都属于此类技术。其特点是以图像拍摄相机参数为媒介，将几何信息与像素形成简单对应和低度融合，在实时渲染中有一定的应用价值。

3) 带显式几何信息，此类技术直接将图像相关几何信息作为输入数据，如像素点对应的三维空间坐标及深度值等。其典型的研究包括：3D image warping[14]，分层深度图像(Layered Depth Images，LDI)[15]渲染和视图相关的纹理映射等方法。带显示几何信息的 IBR 技术是本文的研究重点，该方法强调几何信息深度融合到像素渲染中，已接近传统的图形渲染管线。

1.3 3D image warping

3D image warping 是一种被广泛用于 3D 视频处理[16]和基于图像的渲染技术[17]。在实时三维图形交互式渲染时，该方法常被用来减少交互延迟[1-2,18-19]。当输入图像中的信息不足以在新的视点产生整个图像时，该方法就会产生“孔”失真现象。解决这种问题的方法很多，其中包括分层深度图像[15]，超视觉失真[20]和双重图像变换[18]等。

2 云端预渲染存储

以图片为主要存储信息的各种 IBR 技术也包含各自的存储结构，如：多层广告板技术分别用 KD 树和 BSP 树来组织不同空间内面向相机的视图[21,22]。LDI 树和 Sprite 树则同时使用八叉树来存储相应的结点数据[15,19]等。与 IBR 技术中的存储内容相似，Cloud Baking 在云端染的光影渲染结果被保留在光图中，因此本文构建“光图树”结构以便光图的存储、查找及调用，如图 1 所示。

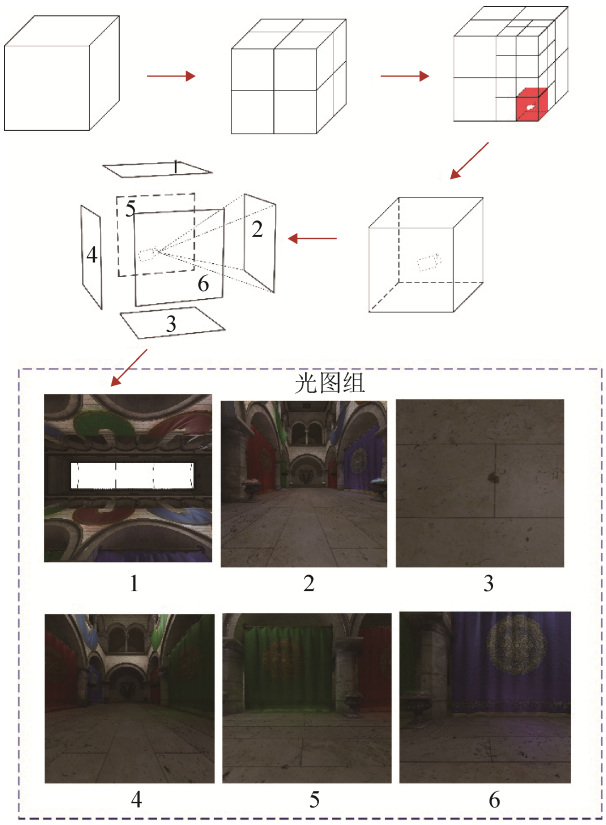


图 1 光图树结构
Fig. 1 GI-Map structure

2.1 叶结点—光图组

在协同式渲染系统中，服务器和客户端的渲染任务同步执行。两者渲染能力的差异性使得它们在处理数据的过程中都宜采用适应各自渲染环境下的数据结构。因此，本文针对服务器端和客户端采用不同的存储结构，而光图组作为子结点被同时应用于 2 个存储结构中。

光图是在同一摄像机在相同的方位下所拍摄的光图(GI map)和深度图(Depth map)的组合，在光图结构中这两个属性分别用 GIMap 和 DepthMap 表达。Direction 和 Position 则描述了拍摄相机的朝向和位置，详细如表 1 所示。光图组是在同一视点不同朝向所建立的一组光图的集合，所拍摄的光图集合力求达到当前视点的全景拍摄结果。如图 1 所示，摄像机以视点为中心，朝着各个方向进行拍摄，本文预设 6 个方向，其他参数信息和前端的摄像机信息保持一致。本文同时设计了光图组结构以保存上述信息，具体如表 2 所示，并且将这一结构作为光图树的结点结构。

表 1 光图结构
Tab. 1 Data Structure of GI-Map

属性名	类型	备注
GIMap	Texture	光图
Direction	Vector	摄像机朝向
Position	Vector	摄像机位置
DepthMap	Texture	深度图

表 2 光图组结构
Tab. 2 Data Structure of GI-Map Group

属性名	类型	备注
SceneIndex	Integer	场景标识索引号
GIMaps	Texture[N]	结点包含的所有光图
Position	Vector	结点的中心位置
Pathcount	Integer	访问结点的次数
Adjacent	GIMaps_group[N]	邻接信息指针数组

2.2 存储结构——光图树

光图树是一颗把场景的三维空间进行有效细分的八叉树，结点中主要包含了在其空间内拍摄的光图组信息，如表 1 所示。而八叉树结点的深度决定了相机拍摄的三维区域，直接表现在图片拍摄的清晰度上。结点深度值越大，相机拍摄的区域越小，光图的清晰度就越高，反之则反。由于大多数 3D 虚拟场景的漫游应用中，场景的入口点相对固定，因此在本文中，预渲染目标选定在初始位置所在的叶结点以及该结点的父结点，具体如见图 1 所示。

2.3 基于结点访问次数的快速查找——热度哈希表

结点的访问次数定义为结点的热度，访问次数越多热度越高。场景中经常被用户访问的结点集合我们称之为热度区域，此部分区域的查找时长是影响系统执行效率的关键因素之一。本机制专门为此构建了比光图树查找效率更高的数据结构—热度哈希表，以达到降低热度区域的查找时间的目的。热度哈希表保存光图树中热度较高的叶结点，哈希表的键是相机的空间位置，键经过散列函数计算出的光图树索引号为数据的存储位置，具体如图 2 所示。

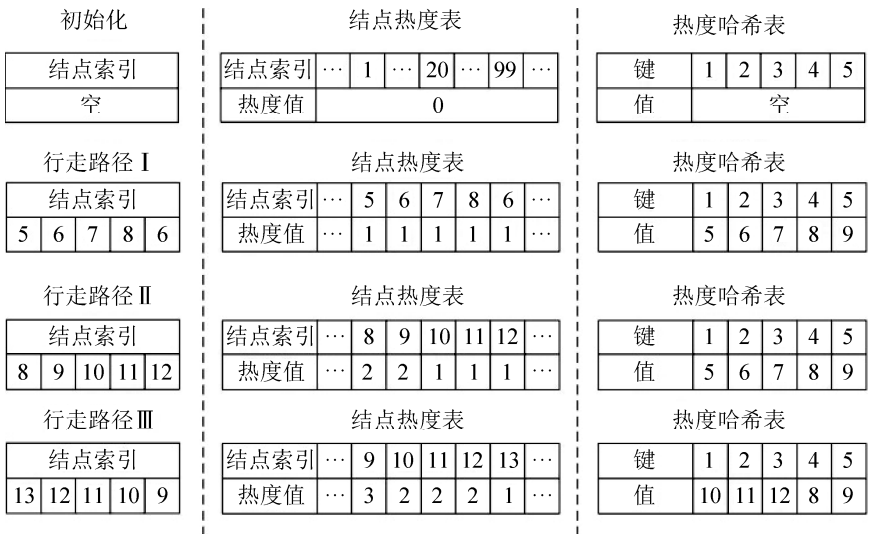


图 2 热度哈希表
Fig. 2 Heat-Hash Table

哈希表的散列函数如式(1)所示：

$$K=\frac{P_x}{L_x}\times 2^{3(d-1)}+\frac{P_y}{L_y}\times 2^{2(d-1)}+\frac{P_z}{L_z}\times 2^{d-1} \quad (1)$$

式中：P_x，P_y，P_z是光图相机在世界空间中位置 P 的 3 个坐标分量；L_x，L_y，L_z是场景 AABB 包围盒在 3 条坐标轴上的长度；C_x，C_y，C_z是光图相机所在结点的三维空间中心位置 C 的 3 个坐标分量；d 是八叉树的深度；K 是热度哈希表的键值。

3 协同式渲染系统的数据调度优化

3.1 用户行为驱动的数据调度构架

Cloud Baking 在加入了预存储结构后，与其对应的数据调度策略也会有一定的变化。当用户在虚拟场景中漫游时，若 Web 前端每帧都向云服务器发出新的请求，无疑会增加云服务器的压力，因而系统将会检测用户的位置与 Web 前端接收结点所处三维空间的关系，并根据用户的行为决定渲染请求。与 Cloud Baking 中的服务器端响应机制相似，光源与相机信息的变换仍然是 Web 前端发出请求的主要条件。所不同的是，移动相机的可视范围仍在同一叶结点中，Web 端是不发送请求的；只有当 Web 前端相机位置到达叶结点所在三维空间的边界附近或相机视域范围超越当前结点所在三维空间时才向云服务器发出请求；后端光图管理器接受到该请求之后会在存储结构中找出对应的结点，并对结点中的光图数据进行 H.264 编码处理之后传输到 Web 前端；Web 前端使用接收到的信息进行 H.264 解码后重建光图，并将其与直接光照图进行混合呈现给用户。详细见图 3。

3.2 云服务器端数据调度

云服务器端的光图管理器在初始状态预存储起始光图树叶结点及父结点。由于父结点可以涵盖 8 个子结点的三维空间，因此当用户驱动 Web 前端摄像机走到子结点的邻接兄弟结点时，如果后端未及时传来兄弟结点的渲染信息，则可以借助父结点的光图预渲染信息进行渲染直到兄弟结点的渲染结果传过来后再更新。实际应用时，本机制设置光图树的叶子结点只有在首次访问或访问时光源发生变换时才能更新对应的光图组。当摄像机的视景体覆盖多个叶结点的范围时，则可以通过传输父结点代替传输多个叶结点。另外，若光图管理器中当前渲染叶结点的光图信息缺失时，则一边让云后端执行当前叶结点的光图渲染，一边让 Web 前端用父结点中的预渲染信息作为当前叶结点的光图以协助 Web 前端渲染；如果叶结点的光图渲完并传到 Web 端后，再用叶结点光图换回父结点预渲染光图。由于父结点中的光图信息覆盖三维范围广，因而光照精细度比实际子结点中的光图信息要低，用父结点取代子结点生成的渲染结果会有像素模糊现象，但与没有光图所造成的全局光照缺失的失真结果相比，效果上是可以接受的，如图 4 所示。

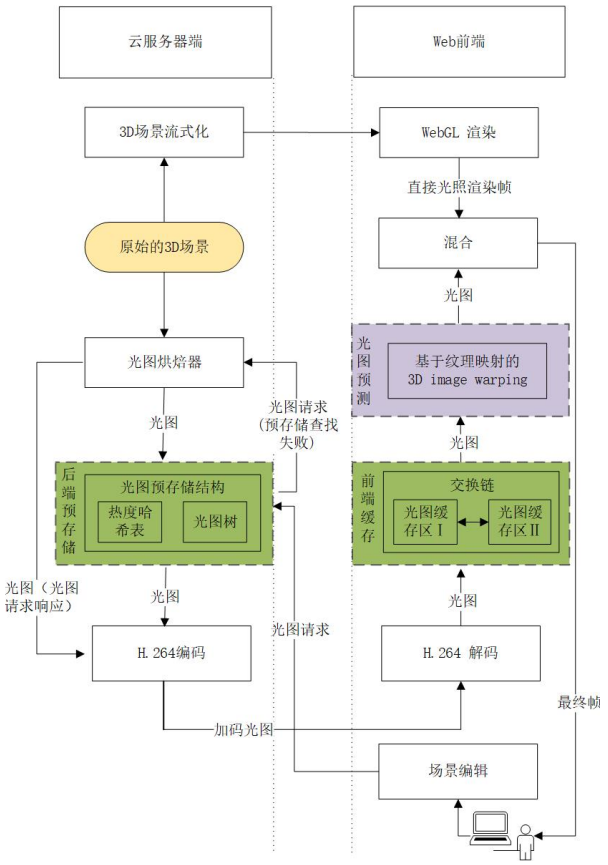


图 3 用户行为驱动的数据调度策略图

Fig. 3 User Behavior-based Data scheduling strategy

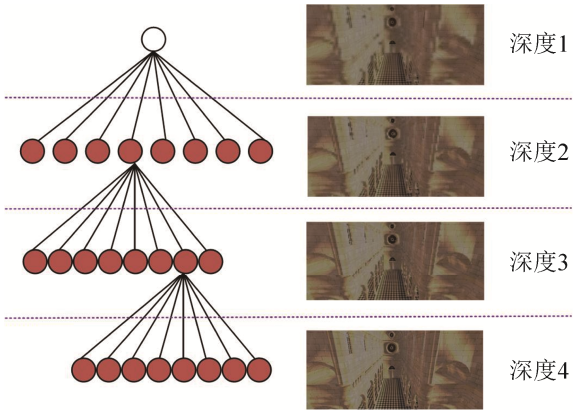


图 4 不同深度结点光图混合后所得最终帧渲染结果
Fig. 4 Final Results of blending GI-Maps of Different Levels

综上，当云服务器后端收到 Web 前端的请求后，后端将会根据前端相机的位置在光图树中定位结点，并将该结点传输至 Web 前端。如果传输结点中光图信息缺失，则将父结点代替子结点传输到前端；如父结点中光图信息缺失，则将父结点的上一级结点代替叶节点传输到前端，以此类推直到找到有图像信息的结点并将其传输到前端。与此同时，云后端将渲染被传输结点的子结点中的光图信息，并层层推进直至渲染完叶结点为止。

3.3 Web 前端的数据调度

Web 浏览器加载能力的局限性导致 Web 前端其无法同时加载大规模云端预渲染数据。在渲染过程中，Cloud Baking 的 Web 前端在使用云端渲染数据后就立刻将之清除，这有利于节省 Web 浏览器的缓存空间，但对于云端渲染资源显然是一种浪费。为此，我们为 Web 前端创建一个光图缓存交换链(Swap Chain)来异步接收结点信息。交换链中有 2 个光图缓存区分别叫当前缓存与待用缓存，一个缓存区将会存储且仅存储一个结点的信息。当前缓存存储的是 Web 前端当前正在使用的结点数据，待用缓存是 Web 前端预先接收将要使用的结点。当摄像机进入新的结点时，则翻转当前缓存与待用缓存。翻转后，待用缓存变为当前缓存并在前端显示之前预存储的，而上一帧的当前缓存则变为待用缓存用来接收新的预使用结点。

3.4 数据查找优化

查找的时长一定程度上影响了云服务器响应的速度，因而本文在上述数据结构基础上对此进行了优化。基于八叉树的查找比传统的遍历查找效率更高，结合热度哈希表查找将进一步大幅度提升查找效率。因此，机制将整个后端的数据查找流程设置为：1) 接受前端请求；2) 热度哈希表中查找，如查找成功则结束查找；3) 如在热度哈希表中查找失败(键值不存在)，则转到光图树中继续查找。此流程相较于在光图树直接查找前增加了一次在热度哈希表中的查找操作，即给了高热度结点一次直接查找命中的机会。为了提升哈希表中的查找命中率，在每一次渲染任务结束后该表中热度值高的结点将会替换热度值低的结点。本文将哈希表中的热度更新策略引入云服务器数据调度算法中，详细请见算法 1。

算法 1 云服务器端调度算法

```
1.  Begin
2.      接受到前端请求；
3.      if 在 Hash 表中找到请求叶子结 then
4.      叶子结点热度值加 1；
5.        发送该叶子结点数据至 Web 前端；
6.  else
7.    将渲染结点设为根结点，传输结点为空；
8.    while 渲染结点有光图信息 do
9.        替换传输结点为渲染结点；
10.       替换渲染结点为渲染结点的子结点；
11.           end while
12.   if 传输结点为叶子结点 then
13.         叶子结点热度值加 1；
14.         发送该叶子结点数据至 Web 前端；
15.     if 结点热度>Hash 表中结点热度 then
17.         用此结点替换 Hash 表中对应结点;
18.         end if
19.       else if 传输结点为空 then
```

```
20.                服务器渲染根结点;
21.                发送根结点数据至 Web 前端;
22.            else
23.                发送结点数据至 Web 前端;
24.                服务器渲染渲染结点并存储;
25.            end if
26.        end if
27.    end
```

4 光图预测

如果要将前端的直接光照渲染帧与后端的光图融合在一起，应保证分别投影这两个图像的相机必须在世界空间中具有相同的方向和位置。然而，Cloud Baking 面临的挑战之一是交互延迟。当用户移动视点时，从客户端向云服务端提出光图请求到云服务端渲染好光图再传回，至少需要一次往返时间。另外，云端光图渲染存在渲染延迟，光图的 H.264 编码也会导致编码延迟。即使具有直接光照的场景可以在本地立即使用新视点渲染，但与新视点对应的光图也将在稍后到达。此延迟会导致前端的直接光照帧和光图变为异步，进而导致混合两组图片的混合产生“重影”失真，如图 5(a)右栏所示。Cloud Baking 中是通过传统的 3D image warping 来进行优化的，这是由于 Cloud Baking 针对的是海量级场景(场景模型数据数量级巨大)所导致。Web 前端有限的加载能力要求海量数据(主要是场景中的模型数据)只有在轻量化后才能在进行加载和显示。然而，轻量化后的模型数据显然与原数据有较大的差别，这些差别造成了 Cloud Baking 只能基于传统 3D image warping 方法来优化云端传来的光图。

而本文主要针对通用网页虚拟现实应用，其场景数据量并不大，完全不需要在前端使用轻量化模型。在使用 Cloud Baking 执行此类应用时，前后端的模型数据保持一致。因此，提出了一种新的 3D image warping 方法，该方法主要通过三维顶点与纹理坐标重新绑定的形式来重建后端传输过来的高质量光照信息，称为 3D texture warping。

(a) 直接混合图

(b) Warping 算法后的混合图

图 5 直接混合与使用 Warping 算法后混合的比较

Fig. 5 Blending result with Warping VS. without it

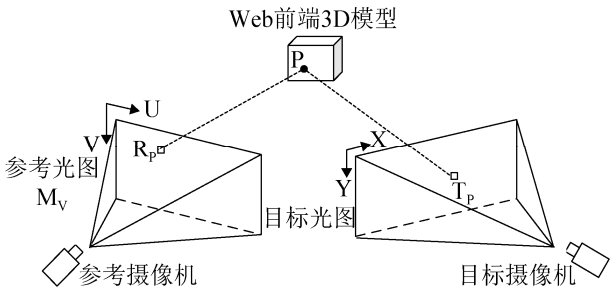
4.1 3D texture warping

传统的 3D image warping 方法分为 3 个步骤：

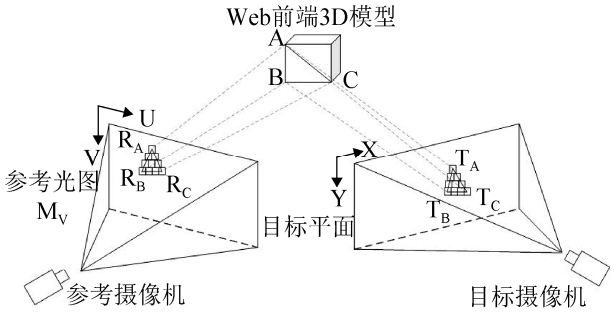
1) 参考图片像素投影，通过将参考图像平面上的像素投影到 3D 空间中形成“具有颜色的三维点云”；

2) 三维点云再投影，这些“具有颜色的三维点云”再投影到目标图像平面上形成一张新图片。这种技术通常被用在 IBR 应用中，以解决摄像机参数偏差所造成的图像错位问题。Cloud Baking 中充分的利用了这一技术，它将后端光图所在平面作为参考图片，把前端渲染出的图像作为目标图像，传递到前端的光图经过 3D warping 之后将完全与前端本地渲染的图像相“贴合”，如图 5(a)所示。面向 WbVR 应用的 Web 前端已知一个重要的额外信息——虚拟场景的 3D 模型的信息，且该信息和后端的模型信息保持一致。因此，本文可以将传统 3D image warping 换成新型 3D image warping——基于纹理的 3D image warping。该方法充分利用 WbVR 应用的特殊性，省去传统 3D image warping 中的最复杂第一个步骤，不再将参考图片的像素投影到三维空间，而

是将该图片作为纹理直接映射到 Web 前端的模型之上。此算法将计算量从逐像素计算变成了关键像素(模型顶点)计算，减少了计算量。而且整个映射过程采用传统的线性纹理过滤，比传统的逐像素过滤效率更高。在实际运用中，本文将 Web 前端接受到的光图中的某一个方向的参考图像(texture)设置成纹理，然后根据已知的 3D 模型的顶点坐标和摄像机的矩阵计算出与之绑定的 UV 纹理坐标。最后，通过求出的纹理坐标把纹理绑定到已知的 3D 模型的顶点上，并将绑定好“新纹理”的 3D 模型的顶点投影到本地渲染摄像机即可，如图 6(b)所示。



(a) 传统 3D image warping



(b) 3D texture warping

图 6 三维图像扭曲

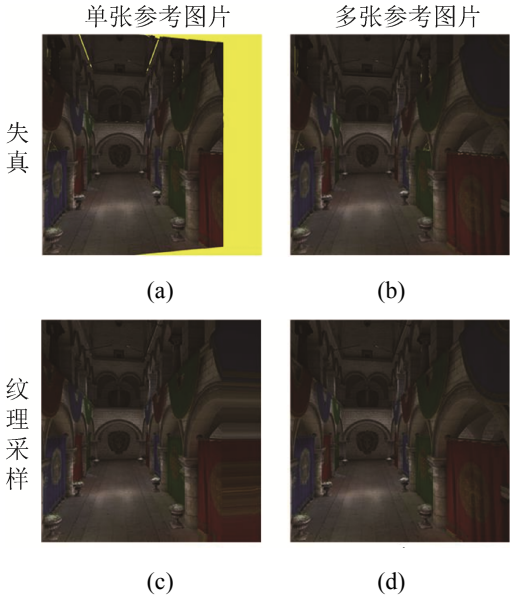
Fig. 6 3D image warping

$$UV = \frac{(M_{Proj} \times M_{View} \times M_{World} \times P).xy}{(M_{Proj} \times M_{View} \times M_{World} \times P).w \times 2.0} + (0.5, 0.5) \quad (2)$$

UV 坐标详细计算如公式(2)所示，其中 P 代表 3D 模型在本地空间(local space)的坐标点，它都包含 x，y，z，w 4 个元素；x，y，z 分别代表相应 3 个坐标的值；w 为齐次值；UV 代表参考图片的纹理坐标值，它是一个二维向量；M_{proj}，M_{view}，M_{word} 分别参考相机对应的投影矩阵，视图矩阵和世界矩阵。

4.2 多张参考图的优化

3D image warping 都会产生伪孔影，其主要原因是参考图像包含的像素信息不够填充新的目标平面所需要的全部信息，本质上是由于拍摄相机的参数差异所导致。本文提出的 3D texture warping 也是一样，并且产生的“孔”失真多以图元形式出现。如果仅以一张光图作为参考图片，失真效果比较明显，请见图 7(a)，图 7(e)的黄色区域。解决这一问题的传统方法就是采样，本文只用单张参考图进行纹理采样时效果有限，如图 7(c)，图 7(g)所示。为此本文提出基于多张参考图片和纹理采样相结合的方法。基于 Web 前端的存储结构，本文基于拍摄相机的参数相似度采集了 3 张光图作为纹理 Warping 的参考图片，用和 Web 前端摄像机相似度最高的光图作为主要采集参考图，另外两张作为补充参考图，只有在主要参考图上找不到纹理坐标时才会使用补充参考图，如图 7(b)，图 7(f)。当补充参考图上也找不到时，则采用纹理采样的方法来补充缺失的像素，如图 7(d)，图 7(h)。



Sponza

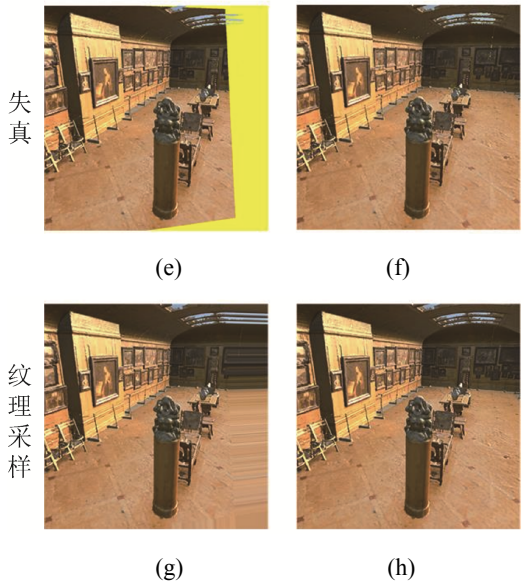


图 7 单张参考图和多张参考图片的 warping 效果比较
Fig. 7 3D Texture Warping Result with One Reference Image VS. with Multiple Images

4.3 基于相机相似度的图像采集

在 Web 前端的数据结构中存储的是当前摄像机所在空间的光图组的多张光图，在每次渲染时需像机和前端摄像机的相似程度来确定光图当前帧的相似度。通过相似度来决定当前深度光图组中作为参考图片和补充图片的光图。相似度最高的作为参考图片，其次则为补充图片。

参考图片的摄像机和前端渲染的摄像机位置和旋转角度的差别越小，则目标图像在重建的过程中缺失的像素越少。因此，本文用光图的摄像机的位置和旋转角度同 Web 前端的摄像机的位置和旋转角度的差异作为评价相似度的标准。该相似度评价算法的输入是光图的摄像机位置和旋转以及 Web 前端的摄像机位置和旋转，输出评价后的相似度值 S 如公式(3)所示，其中 P_0，P_1 分别是 2 个用来比较的光图相机的位置；R_0，R_1 是上述 2 个相机的旋转角度；Dist(,)是位置间距的函数；Dot(,)是向量点乘函数；L_{BBD} 是 2 个相机所在结点三维空间 AABB 包围盒的对角线长度。

$$S=\frac{\mathrm{Dist}\ (P_0\ ,\ P_1)}{L_{BBD}}+\frac{Dot(\ R_0\ ,\ R_1)}{\|\ R_0\ \|\ \bullet\ \|\ R_1\ \|} \tag{3}$$

5 实验

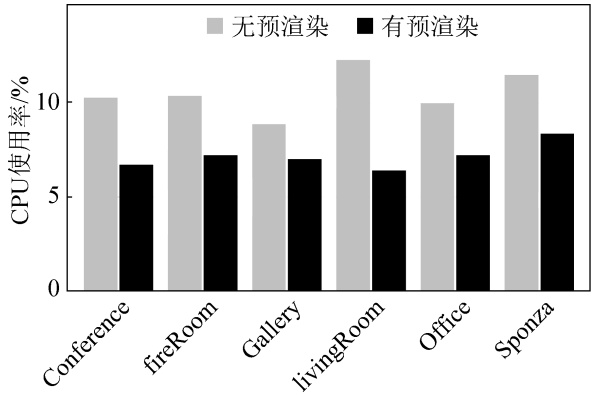
本文在 Cloud Baking 基础上引入光图树渲染机制，测试前端 PC 硬件配置：CPU 为 I7-7700HQ，GPU 为 GTX1060M，内存为 8G；测试云服务器硬件配置：CPU 为 Intel Xeon E5 2640X3 2.6 GHz，GPU 为 Nvidia Quadro M6000，内存为 128 GB。前端 PC 操作系统为 Windows 10，后端服务器端操作系统为 Windows Server 2012。所使用的 Web 浏览器是 Google Chrome，版本号是 61。为了验证光图树渲染机制在协同式渲染中发挥的作用，我们完成了以下 4 组测试实验。

5.1 预渲染机制优化下的 Cloud Baking 性能测试

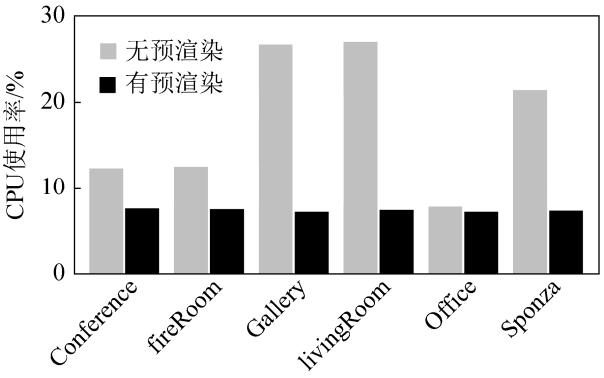
实验 1 测试了加入预渲染机制后的 Cloud Baking 云端服务器 GPU 和 CPU 占有率，并与原始的 Cloud Baking 进行比较。为了测试力求测试的准确性，挑选了 5 个典型测试场景，如表 3 所示，并要求 Web 前端在整个渲染过程中不断发起渲染请求。测试结果表明，预渲染机制优化下的渲染系统比之原系统有明显偏低的 CPU 和 GPU 的使用率(占有率)，如图 8 所示。由此说明加入预渲染机制的 Cloud Baking 极大的降低了对硬件资源的需求，提升了渲染的效率，是非常行之有效的优化手段。

表 3 测试模型表
Tab. 3 3D Test Models

模型名称	面片个数	数据量/MB
conference	331,179	19.9
fireplace	143,173	20.9
gallery	998,941	71.4
livingRoom	580,647	40.8
office	10,033	7.76
sponza	262,267	103.0



(a) 有无预渲染情况下的系统 CPU 使用率比较图



(b) 有无预渲染情况下的 GPU 使用率比较图

图 8 有无预渲染情况下 CPU//GPU 使用率对比图
Fig. 8 CPU/GPU Load Rate With Pre-Rendering VS. without It

5.2 预渲染机制优化下的 Cloud Baking 性能测试

光图的预存储的应用从本质上来看是云渲染服务器用存储空间换取渲染时间的节省并提升渲染效率的一种方法，然而空间的提升也是一种资源的消耗，如何在时间和空间中取得一定的平衡是实验 2 的主要目的。在该实验中，假定查找时间与渲染时间相比可以忽略不计，即每调用一次光图预存储结构中的光图数据则节省一帧渲染时间。我们设定云端渲染帧率为24 fps，即每帧的渲染时长为 1/24 s。因而，我们统计出光图树结点被调用的次数，将次数与帧渲染时长的乘积作为节省时长，最后用节省时长与程序运行时间比值作为整个应用的时间节省率。X 轴为测试的存储空间，分别为装载光图后，深度为 2，3，4，5 层的光图树的存储空间大小。Y 轴为其对应的时间节省率，测试场景为 Sponza。我们发现对于云渲染器来说，当光图树的深度为 4 时，所耗存储空间大小为900 M且世界节省率达到50%，是比较理想的光图树设置深度。

5.3 热度哈希表对光图树查找的影响

本文中的光图树结构引入的热度哈希表，因而实验 3 将原光图树与带热度哈希表的光图树查找时间进行比较。为保证测试的通用性，本文随机生成了 64 000 000 个的三维空间位置(由于数据量巨大，未在以上光图树结点中存放光图，仅放入相关位置信息)作为测试数据集，并将其中一半的测试数据设置为热度区域的三维空间位置。另外，哈希表长度被为叶子结点数的 1/5。在此基础上，我们对比了深度从3~7之间的5棵光图树，如图 9 所示。结果表明，在查找耗时上带热度哈希表的光图树明显比不带光图树的耗时更短，且随着深度值的增加这一差距进一步的扩大，因而以用户行为作为优先考虑的热度哈希表的引入明显的提升了查找效率。

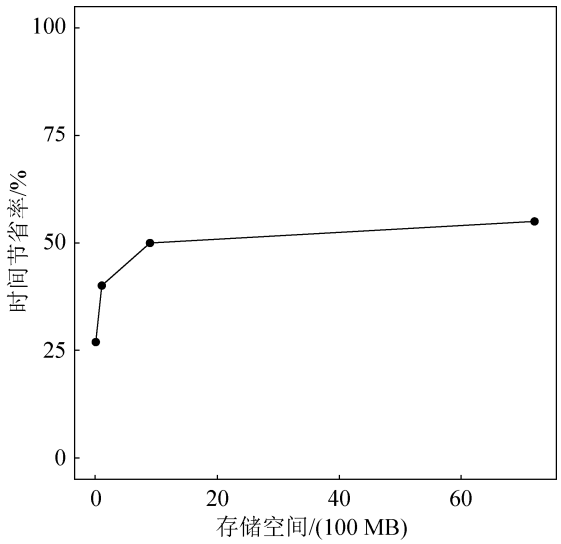


图 9 时间节省率与光图树存储空间消耗对比
Fig. 9 Time Save Rate VS. Storage of GI-Tree

5.4 3D texture warping 的性能测试

实验 4 测试了本文中所设计的 3D texture

warping 性能，并与传统的 3D image warping 进行比较。同样，为保证测试的准确性与相关实验的延续性，测试工作仍然以上述五个场景为测试对象，在整个渲染过程中分别引入不同 2 种 3D image warping 来优化渲染结果。测试结果表明，基于新型 3D image warping 的渲染系统比之用传统 3D image warping 的渲染系统也同样有着明显偏低的 CPU 和 GPU 的占有率，如图 10 所示。这进一步说明 3D texture warping 无论在性能表现还是在渲染需求方面都是优于传统的 3D image warping。

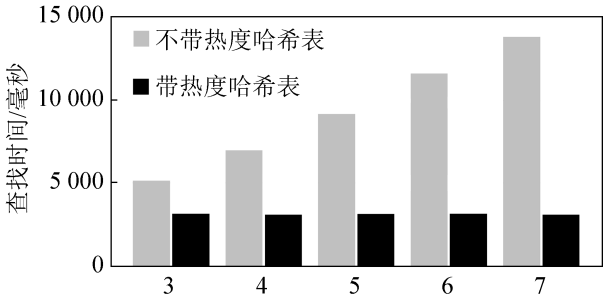


图 10 光图树查找时间对比
Fig. 10 Search Time of Light Tree

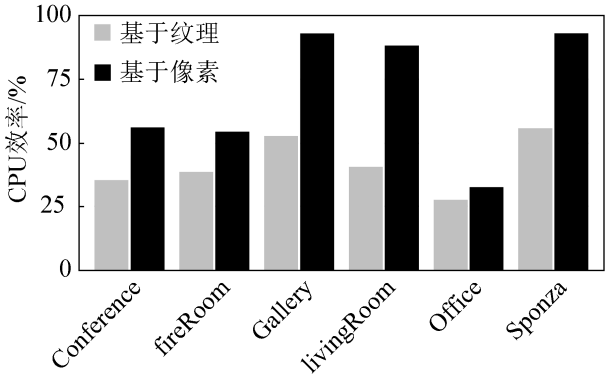
5.5 基于多图纹理映射的 3D image warping 的光图预测渲染效果测试

实验 5 研究了在多张参考图片优化下 3D texture warping 对于渲染效果的影响。本组实验将我们的方案与另外两种方案做对比，这 2 种方案分别是：

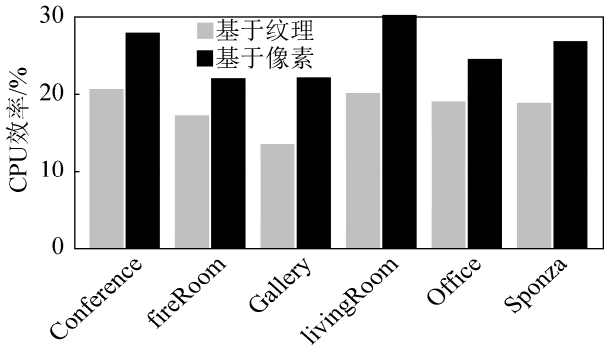
1) 传统的 3D image warping 方案，即当 Web 前端渲染时发现光图相机与前端相机不一致，则用 3D image warping 来优化。

2) 光图复制方案，即在上述方案的基础上，不做 3D image warping 只是简单地复制最后接收光图以解决由延时导致前后端相机不一致的问题。这种方法如图 5 所示，当我们从前后两个不同的视点用复制光图法混合直接光照帧和光图时，将导致严重的重影失真。

设 M_V 是来自视点 V 的正确光图，M_V'是视点 V 处预测出的光图；I_V 和 I_V'分别是前端的直接光照图片与 M_V 和 M_V'混合所得图像。为了客观地评估光图的质量，我们使用 SSIM，并绘制了帧率从 4fps 到 28fps 的条件下，I_V 与 3 种方案下的 I_V'比较所得出的 SSIM 的平均值，如图 11 所示。



(a) 不同 warping 机制下系统 CPU 使用率比较



(b) 不同 warping 机制下系统 GPU 使用率比较

图 11 不同 warping 机制下 CPU/GPU 使用率对比
Fig. 11 Utilization ratio comparison of CPU/GPU In Different 3D Warping Conditions

图 12 中 x 轴是光图流的帧速率。对于此实验，Web 渲染器渲染帧速率为 60 fps。因此，光图流的帧速率为 20 fps 意味着在两个实际光图之间存在重复的光图。图中可以看到，我们的基于多图纹理映射的 3D image warping 算法可以在多数情况下获得更好的 SSIM 值，意味着比较前两种方法都有一定的提升。本实验参考 Zinner 等[23]的建议，当 SSIM 值为 0.88 时，主观映像评测(mean opinion score，MOS)分数达到 3~5 分之间。因此我们将 SSIM 阈值定为 0.88，由图 12 可知本方法即使在帧率为 4 fps 时，SSIM 值大多超过 0.88，光图质量是令人满意的。

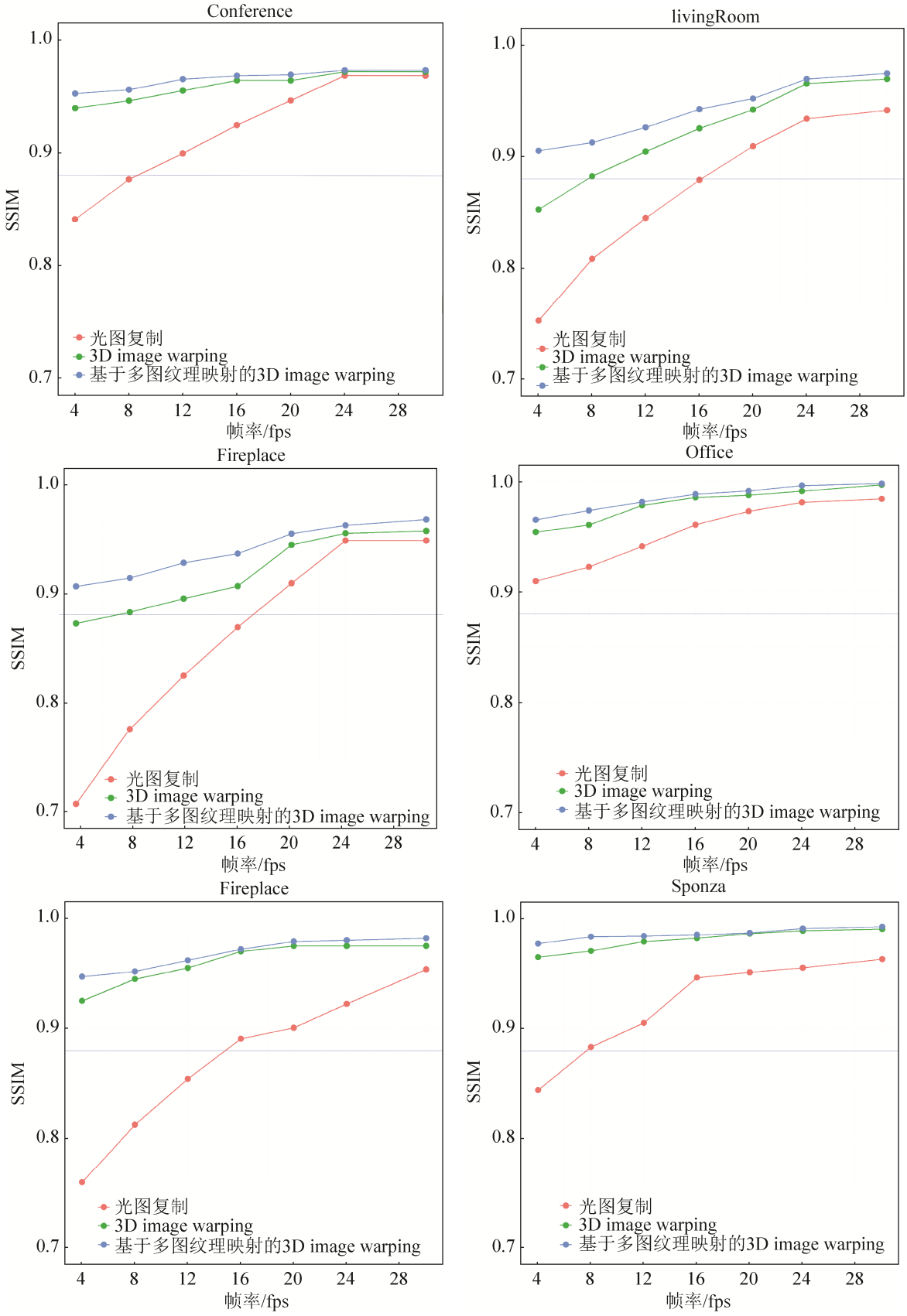


图 12 3 种光图预测之间 SSIM 对比图
Fig. 12 SSIM Contrast Map Between Three Optical Map Predictions

https://dc-china-simulation.researchcommons.org/journal/vol31/iss8/13
DOI: 10.16182/j.issn1004731x.joss.18-VR0731

6 结论

本文以协同协同式实时渲染系统 Cloud Baking 作为研究对象，提出光图树渲染机制，该机制主要包括：

1) 光图树存储结构，用光图树来存储整个场景的渲染信息。

2) 前后端相匹配的调度策略，该调度策略可以快速的找到前端所请求的光图组，重新组织成深度光图组。

3) 基于纹理映射 3D image warping，该方法进一步优化渲染质量和前端运行速度。经过测试表明，经过光图树渲染机制优化后的 Cloud Baking 与未优化之前的系统相比：CPU 和 GPU 负载更低，运行效率更高，拥有更好的渲染效果且更具扩展性。

随着人工智能的发展，将用户的行为数据引入本系统，并基于机器学习以及深度学习的方法进行预渲染操作是项目的下一个目标。目前，已经调研了部分用户的行为数据，包括：用户行进位置、进行朝向以及关注点等，未来会以这些技术为基础展开基于 AI 技术的预渲染渲染机制的研究。

参考文献:

[1] Liu C, Ooi W T, Jia J, et al. Cloud Baking: Collaborative Scene Illumination for Dynamic Web3D Scenes[J]. ACM Transactions on Multimedia Computing, Communications, and Applications (S1551-6857), 2018, 14(3S): 59.

[2] Liu C, Jia J, Zhang Q, et al. Lightweight WebSIM Rendering Framework Based on Cloud-Baking[C]. Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. Singapore: ACM, 2017: 221-229.

[3] Hu Q, Yu D, Wang S, et al. Hybrid three-dimensional representation based on panoramic images and three-dimensional models for a virtual museum: Data collection, model, and visualization[J]. Information Visualization(S1473-8716), 2017, 16(2): 126-138.

[4] Shi S, Hsu C H. A survey of interactive remote rendering systems[J]. ACM Computing Surveys (S0360-0300), 2015, 47(4): 57.

[5] Cai W, Shea R, Huang C Y, et al. A Survey on Cloud Gaming: Future of Computer Games[J]. IEEE Access (S2169-3536), 2016, 4: 7605-7620.

[6] Chen Y, Liu E S. A path-assisted dead reckoning algorithm for distributed virtual environments[C]. IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT). Cosenza, Italy: IEEE, 2015: 108-111.

[7] Pharr M, Jakob W, Humphreys G. Physically based rendering: From theory to implementation[M]. London, UK: Morgan Kaufmann, 2016.

[8] Guthe S, Schardt P, Goesele M, et al. Ghosting and popping detection for image-based rendering[C]. 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2016. Hamburg, Germany: IEEE, 2016: 1-4.

[9] Shi L, Huang F C, Lopes W, et al. Near-eye light field holographic rendering with spherical waves for wide field of view interactive 3d computer graphics[J]. ACM Transactions on Graphics (S0730-0301), 2017, 36(6): 236.

[10] Cohen M F, Szeliski R. Lumigraph[M]. Computer Vision. Boston, MA: Springer, 2014: 462-467.

[11] Pujades S, Devernay F, Goldluecke B. Bayesian view synthesis and image-based rendering principles[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Columbus, Ohio: IEEE, 2014: 3906-3913.

[12] Zhang Y X, Zhu Z Q, Ma P F. Walk through a Museum with Binocular Stereo Effect and Spherical Panorama Views[C]. Culture and Computing (Culture and Computing), 2017 International Conference. Kyoto, Japan: IEEE, 2017: 20-23.

[13] Cao C, Wu H, Weng Y, et al. Real-time facial animation with image-based dynamic avatars[J]. ACM Transactions on Graphics (S0730-0301), 2016, 35(4): 126.

[14] Jin J, Wang A, Zhao Y, et al. A fast region-level 3D-warping method for depth-image-based rendering[C]. Multimedia Signal Processing (MMSP), 2015 IEEE 17th International Workshop. Xiamen, China: IEEE, 2015: 1-6.

[15] Daribo I, Saito H. A novel inpainting-based layered depth video for 3DTV[J]. IEEE Transactions on Broadcasting (S0018-9316), 2011, 57(2): 533-541.

[16] Smolic A, Muller K, Dix K, et al. Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems[C]. 2008 15th IEEE

International Conference on Image Processing (Icip 2008). San Diego, California: IEEE, 2008: 2448-2451.

[17] Fehn C. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV[C]. Stereoscopic Displays and Virtual Reality Systems XI. California, USA: SPIE, 2004, 5291: 93-105.

[18] Shi S, Nahrstedt K, Campbell R. A real-time remote rendering system for interactive mobile graphics[J]. ACM Transactions on Multimedia Computing, Communications, and Applications (S1551-6857), 2012, 8(3S): 46.

[19] Zhu M, Morin G, Carville V, et al. Sprite tree: an efficient image-based representation for networked virtual environments[J]. The Visual Computer(S0178-2789), 2017, 33(11): 1385-1402.

[20] Bao P, Gourlay D. Remote walkthrough over mobile networks using 3-D image warping and streaming[J]. IEE Proceedings-Vision, Image and Signal Processing (S1359-7108), 2004, 151(4): 329-336.

[21] Xu H, Gossett N, Chen B. Knowledge and heuristic-based modeling of laser-scanned trees[J]. ACM Transactions on Graphics (S0730-0301), 2007, 26(4): 19.

[22] Zhang Q L, Pang M Y. A survey of modeling and rendering trees[C]. International Conference on Technologies for E-Learning and Digital Entertainment. Berlin, Heidelberg: Springer, 2008: 757-764.

[23] Zinner T, Hohlfeld O, Abboud O, et al. Impact of frame rate and resolution on objective QoE metrics[C]. The second international workshop on quality of multimedia experience (QoMEX) . Trondheim, Norway: IEEE, 2010: 29-34.

DOI: 10.16182/j.issn1004731x.joss.18-VR0731