无预处理的实时全局光照渲染方法综述

陈拓, 周子恒, 吴震宇, 张松海*

(清华大学计算机科学与技术系　北京　100084)
(shz@tsinghua.edu.cn)

摘　要: 实时全局光照渲染在虚拟现实、增强现实、影视制作、电子游戏等领域中都有重要作用, 是真实感渲染的重要研究领域. 随着硬件算力的不断提高, 实时全局光照渲染领域中涌现出很多新的方法. 文中对无预处理的实时全局光照渲染的相关工作进行综述, 将实时全局光照渲染方法分为3类, 从多个角度进行了分析. 首先概述渲染方程与实时全局光照渲染方法; 然后介绍光栅化、光线追踪和探针法 3 类实时全局光照方法, 并对每类的方法进行分析对比; 最后总结实时全局光照渲染的研究进展, 分析现有方法存在的优缺点, 指出未来可能的研究方向, 包括优化光线追踪采样、神经网络辅助光线追踪、优化探针纹理数据获取等.

关键词: 计算机图形学; 实时全局光照渲染; 光栅化; 光线追踪; 光照探针
中图法分类号: TP391.41　　**DOI:** 10.3724/SP.J.1089.2024-00683

Overview of Real-Time Global Illumination Rendering Methods without Pre-Processing

Chen Tuo, Zhou Ziheng, Wu Zhenyu, and Zhang Songhai*

(*Department of Computer Science and Technology, Tsinghua University, Beijing*　100084)

Abstract: Real-time global illumination rendering plays a crucial role in fields such as virtual reality, augmented reality, film production, and video games, and is an important area of research in realistic rendering. With the continuous improvement of hardware computing power, new methods have emerged in the field of real-time global illumination rendering. This paper reviews the related work on real-time global illumination rendering without pre-processing and categorizes the methods into three types for analysis and comparison. First, we overview the rendering equation and real-time global illumination rendering methods. Then, we introduce and compare three types of real-time global illumination methods: rasterization, ray tracing, and probe-based methods. Finally, we summarize the research progress in real-time global illumination rendering and point out potential future research directions, including optimizing ray tracing sampling, neural network-assisted ray tracing, and optimizing probe-based ray tracing.

Key words: computer graphics; real-time global illumination rendering; rasterization; ray tracing; light probe

真实感渲染始终是计算机图形学重要的研究主题, 其目标是用尽可能少的时间对现实世界的光照完成尽可能真实的模拟. 全局光照对渲染的真实性至关重要, 而实时渲染意味着要在满足实时性(≥30 帧/s)要求的情况下完成渲染, 因此需要对时间和效果作权衡取舍. 在电子游戏、虚拟现

收稿日期: 2024-11-09; 修回日期: 2025-03-11. **陈拓**(2001—), 男, 博士研究生, 主要研究方向为计算机图形学; **周子恒**(2004—), 男, 在校学生, 主要研究方向为计算机图形学; **吴震宇**(2004—), 男, 在校学生, 主要研究方向为计算机图形学; **张松海**(1978—), 男, 博士, 副教授, 博士生导师, CCF 会员, 论文通信作者, 主要研究方向为计算机图形学.

实、增强现实以及影视制作中, 实时渲染技术都得到了广泛的应用.

实时全局光照是一个宏大的话题. 预处理对于实时渲染而言是重要的减少计算量的手段, 如预计算辐射度传输(precomputed radiance transfer, PRT)[1]. 本文仅讨论不作预处理的算法, 且仅考虑对三角网格的渲染, 将实时渲染的技术路线分为光栅化、光线追踪和光照探针 3 类; 同时, 基于神经网络的渲染方法中对于模型的预训练未被归于预处理的概念中.

光栅化通过矩阵变换将场景物体从世界空间变换到屏幕空间, 并且进行光栅化、深度测试等步骤, 确认像素与物体之间的对应关系, 渲染速度更快, 但难以模拟间接光照. 光线追踪则基于光路可逆的思想, 从相机出发追踪光线的传播过程, 以此考虑光线的颜色, 渲染效果更真实, 但由于涉及场景求交, 其速度比光栅化更慢, 多用于离线渲染.

在离线渲染研究中, 由于未进行实时性约束, 因此能够采用一些更精确的渲染方法. 基于辐射度量学, Cohen 等[2]提出的辐射度方法能够精确地模拟室内场景的间接光照, 其思想在实时渲染方法中多有体现, 但由于计算量随面元数量增加呈二次增长, 因此难以直接应用于实时渲染. 实际上, 光照探针可以看作辐射度方法在实时渲染中简化后的变种. 探针法全局光照通过对辐射度等光学物理量的缓存来减少计算开销, 同时使用蒙特卡罗光线追踪的计算框架避免辐射度方程的迭代求解, 从而满足实时要求.

2018 年, NVIDIA 推出 RTX 系列显卡支持硬件光线追踪, 使得光线追踪大大加速. 由于光线追踪的计算量较大, 尤其对于大规模 3D 场景, 因此光线追踪渲染方法在此之前往往难以满足实时性要求[3]. 此后, 光线追踪类算法在实时渲染中得到越来越广泛的应用, 而探针法全局光照作为一种比光线追踪更加快速的方法, 也得到了进一步的研究. 近年来, 硬件对神经网络的支持也让神经网络在实时全局光照渲染中的应用成为研究热点.

本文将从如下方面介绍实时全局光照渲染方法:

(1) 光栅化. 速度最快, 在实时渲染中应用最广泛, 渲染较简单的光照效果;

(2) 光线追踪. 速度较慢, 在实时渲染应用中需要硬件支持, 有较真实的光照效果;

(3) 探针法. 速度与效果均在光栅化与光线追踪之间, 其中, 辐照度探针多用于漫反射表面的全局光照.

本文对目前的实时全局光照渲染方法进行综述, 按照渲染技术路径分为 3 类, 并进行分析对比. 首先介绍实时全局光照渲染方法的理论基础与分类, 然后对光栅化、光线追踪和探针法 3 类渲染方法进行综述, 最后总结分析现有方法的优缺点以及未来可能的研究方向, 以启发后续的研究.

1 实时全局光照渲染方法概述

1986 年, Kajiya[4]提出的渲染方程中使用辐射度量学相关概念, 将渲染问题转化为一个积分求解的问题, 为后续渲染技术的发展提供了坚实的数学基础, 并衍生出全局光照的概念. 在渲染方程中, 渲染的问题被表达为求解某个场景点 p 向某个方向 ω_{o} 发出光线的能量, 即辐射度

$$L_{\mathrm{o}}(p,\omega_{\mathrm{o}})=L_{\mathrm{e}}(p,\omega_{\mathrm{o}})+\int_{\Omega+}L_{\mathrm{i}}(p,\omega_{\mathrm{i}})f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})(\boldsymbol{n}\cdot\omega_{\mathrm{i}})\mathrm{d}\omega_{\mathrm{i}} \tag{1}$$

在光线传播过程中, 辐射度保持不变. 其中, $L_{\mathrm{e}}(p,\omega_{\mathrm{o}})$ 为点 p 向 ω_{o} 方向的自发光项; 考虑以 p 的法线 $\boldsymbol{n}$ 为轴线的单位半球面 $\Omega+$, 半球面上一点对应一个方向向量, 对该半球面方向进行积分; 入射方向为 ω_{i}, 则 $f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})$ 为双向反射分布函数(bidirectional reflectance distribution function, BRDF); $L_{\mathrm{i}}(p,\omega_{\mathrm{i}})$ 为 p 从 ω_{i} 方向接收到的光线的辐射度. p 接收到的光线可以分为 2 部分, 分别是由光源直接出射, 即直接光照部分; 以及由其他场景点反射, 即间接光照部分.

在实际渲染时, 由于遮挡测试的开销较大, 得到的 $L_{\mathrm{i}}(p,\omega_{\mathrm{i}})$ 往往是考虑可见性之前的光线辐射度, 即这根光线可能被其他物体遮挡. 因此, 此时, 需要额外考虑一个可见性项 $V(p,\omega_{\mathrm{i}})$, 即该光线是否能够照射到 p. 渲染方程更改为

$$L_{\mathrm{o}}(p,\omega_{\mathrm{o}})=L_{\mathrm{e}}(p,\omega_{\mathrm{o}})+\int_{\Omega+}L_{\mathrm{i}}(p,\omega_{\mathrm{i}})f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})V(p,\omega_{\mathrm{i}})(\boldsymbol{n}\cdot\omega_{\mathrm{i}})\mathrm{d}\omega_{\mathrm{i}}.$$

全局光照渲染指考虑物体受到的直接光照和间接光照. 直接光照指物体直接受到光源照射产生的光照效果, 间接光照则是光源发出的光线经过场景的多次弹射(包括反射、折射、散射)对物体产生的光照效果, 因此, 间接光照能够更好地模拟现实世界的真实光线传播过程, 对渲染的真实感有至关重要的作用. 然而, 由于算力资源有限, 尤其在动态场景下, 目前的渲染算法难以实时地计

算出光线无限次弹射的光照效果, 因此往往只能考虑前若干层的间接光照. 本文将实时全局光照渲染方法分为 3 类.

(1) 光栅化全局光照. 分为 3D 空间全局光照和屏幕空间全局光照, 其中, 3D 空间全局光照方法通过分析世界空间中的信息完成光照计算, 而屏幕空间全局光照方法则只通过相机光栅化得到的结果进行全局光照计算. 因此, 前者的准确性更高, 但计算开销更大.

(2) 光线追踪全局光照. 多用于离线渲染, 多进行一层光线追踪意味着多考虑一层间接光照. 因此, 光线追踪算法是更贴近真实的渲染方法. 光线追踪使用重要性采样对渲染方程式(1)中的积分进行蒙特卡罗估计, 即

$$\int_{\Omega+} L_{\mathrm{i}}\left(p,\omega_{\mathrm{i}}\right) f_{\mathrm{r}}\left(p,\omega_{\mathrm{i}},w_{\mathrm{o}}\right)\left(\boldsymbol{n}\cdot\omega_{\mathrm{i}}\right)\mathrm{d}\omega_{\mathrm{i}} \approx \frac{1}{N}\sum_{j=1}^{N}\frac{L_{\mathrm{i}}\left(p,\omega_{ij}\right) f_{\mathrm{r}}\left(p,\omega_{ij},w_{\mathrm{o}}\right)\left(\boldsymbol{n}\cdot\omega_{ij}\right)}{\mathrm{PDF}\left(\omega_{ij}\right)}.$$

其中, N 表示所采样的光线数量; p 采用某种采样策略在以法线 $\boldsymbol{n}$ 为轴线的半球面方向进行采样; 所采样的第 j 根入射光线方向为 ω_{ij}; $\mathrm{PDF}\left(\omega_{\mathrm{i}}\right)$ 表示采样到该方向的概率密度函数(probability density function, PDF)值. 当 $L_{\mathrm{i}}\left(p,\omega_{\mathrm{i}}\right) f_{\mathrm{r}}\left(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}}\right)\left(n\cdot\omega_{\mathrm{i}}\right)$ 与 $\mathrm{PDF}\left(\omega_{\mathrm{i}}\right)$ 成正比时, 上述估计的方差取到最小值 0.

实时渲染中, 受到采样数的限制会产生噪点问题, 需要使用降噪等手段优化渲染结果. 近年来, 在实时光线追踪中使用神经网络成为了研究热点.

(3) 探针法全局光照. 光照探针是一个虚拟概念, 仅表示一个空间位置, 没有空间实体. 在硬件算力一定的情况下, 将光线追踪的对象从像素变为数量级更小的探针时, 允许对每个探针追踪更多根光线, 计算探针周围的局部光照信息. 最后使用光栅化渲染, 在渲染像素颜色时获取像素对应场景点周围的探针信息, 用于估计间接光照.

上述实时全局光照渲染方法的提出时间如图 1 所示.

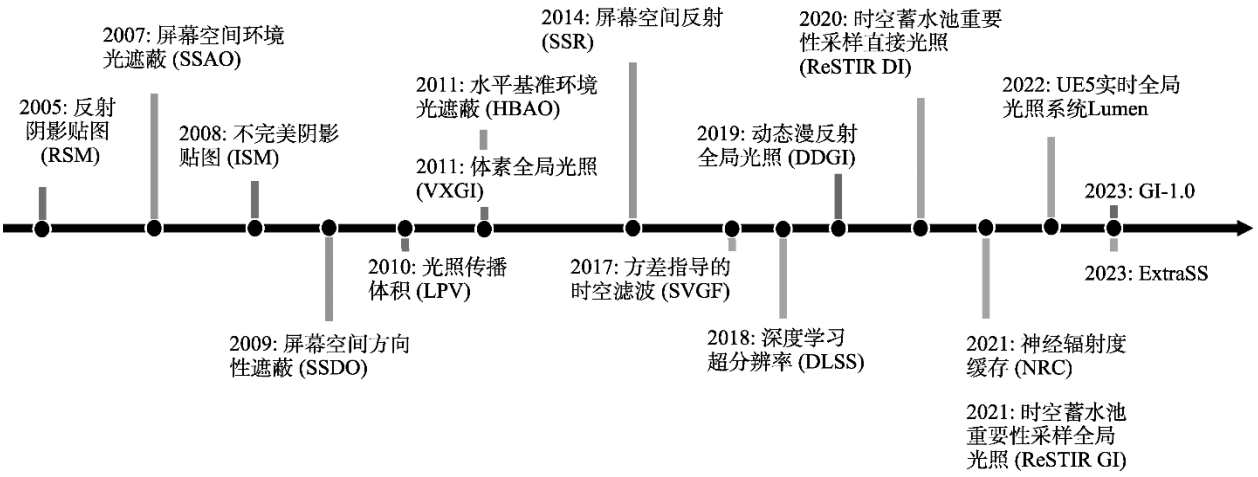

图 1　实时全局光照渲染方法时间线

2　光栅化实时全局光照

光栅化实时全局光照基于现代渲染流水线, 在顶点着色器中完成对顶点坐标的投影, 并在像素着色器中进行着色. 与光线追踪相比, 光栅化实时全局光照是一类更快速、应用也最广泛的全局光照算法. 依据是否仅使用直接光照渲染得到的屏幕信息, 光栅化实时全局光照可以分为 3D 空间全局光照和屏幕空间全局光照. 其中, 前者往往更精确; 后者由于仅使用屏幕信息, 难以考虑屏幕外的信息, 因此在反射时会出现对隐藏几何体渲染错误的情况.

下面分别介绍并对比 3D 空间全局光照与屏幕空间全局光照的经典算法.

2.1　3D 空间全局光照

3D 空间全局光照的经典算法包括反射阴影贴图(reflective shadow maps, RSM)算法、光照传播体积(light propagation volumes, LPV)算法和体素全局光照(voxel global illumination, VXGI)算法, 它们各有对应的改进算法.

2.1.1　RSM 算法

RSM 算法由 Dachsbacher 等[5]提出, 是一种经典的 3D 空间全局光照算法, 由于物体被光线照射后会反射光线, 因此将其视为新的次级光源继续

照亮周围的其他物体. 该算法首先正常计算直接光照的影响, 然后从光源的角度进行一次渲染, 得到的深度贴图称为阴影贴图(shadow map, SM), 其中每个纹素都作为虚拟点光源(virtual point light, VPL)参与间接光照计算. 虚拟点光源 x_{p} 对屏幕像素对应的着色点 x 贡献的辐照度公式为

$$E_{\mathrm{p}}(x,\boldsymbol{n})=\boldsymbol{\Phi}_{\mathrm{p}}\frac{\max\left\{0,\left\langle \boldsymbol{n}_{\mathrm{p}}\mid x-x_{\mathrm{p}}\right\rangle\right\}\max\left\{0,\left\langle \boldsymbol{n}\mid x_{\mathrm{p}}-x\right\rangle\right\}}{\left\|x-x_{\mathrm{p}}\right\|^{4}}.$$

不同 VPL 对该像素的贡献总和简单相加即可. 为了进一步减小这一过程的计算量, RSM 做如下重要假设.

(1) 尽管 x 和 x_{p} 之间可能有物体遮挡(则 x 应不可被照亮), 但始终认为 x 可以被 x_{p} 照亮. 该假设会引入渲染结果的误差, 但往往在可接受的范围之内.

(2) 通过随机采样进一步减少要考虑的 VPL 数量. 将 x 投影到光源坐标, 得到 SM 上的 2D 坐标 (s,t); 之后只考虑以这个点为圆心、$r_{\max}$ 为半径的圆内的 VPL, 生成一个 2D 随机数 (ξ_1,ξ_2), 并对 SM 上坐标为 $\left(s+r_{\max}\xi_1\sin(2\pi\xi_2),t+r_{\max}\xi_1\cos(2\pi\xi_2)\right)$ 的点对应的 VPL 进行采样. 由于距离投影点较远处采样密度较小, 因此对采样得到的反射辐照度赋以权重 ξ_1^2, 最后进行归一化以保证能量守恒.

RSM 的实现虽然较为简单, 但需要为每个光源渲染 SM, 同时做了忽略可见性的假设, 因此其性能受到光源数量与场景复杂度的限制; 同时, RSM 仅考虑了一层间接光照, 并没有考虑光线在场景中的更多次弹射.

针对上述缺陷, Schied 等[6]提出一种着色方法, 只需要一组较小的 SM, 但选择这组光源的过程不能实时在线完成; Kaplanyan[7]则提出一种对 RSM 的分割预处理手段, 从而提取一组虚拟区域光源用于着色计算; 针对 RSM 仅能估计一层间接光照且无法考虑可见性的问题, Ritschel 等[8]引入不完美阴影映射(imperfect shadow maps, ISM)表示每个 VPL 的低分辨率缓冲区, 使用一组点作为场景的粗略近似. ISM 用于间接光遮挡, 并且可以扩展到不完美反射阴影映射获取光的多次反射. Ritschel 等[9]在选择 RSM 像素的过程中使用屏幕空间信息, 进一步改进遮挡检测; 吴向阳等[10]通过引入形状因子对 RSM 的精度进行了优化. 与 RSM 的思路类似, 徐翔等[11]提出基于点的全局光照绘制方法对直接光照点进行存储, 并且使用这些点进行间接光照计算.

2.1.2 LPV 算法

2009 年, Kaplanyan[7]提出 LPV 算法, 将场景均匀划分为体素, 使用相机的深度缓冲区、几何缓冲区和 RSM 的信息, 在体素中存储和传播光照的辐射度, 并使用场景点对应的体素格子的辐射度计算间接光照. 其算法步骤如下.

Step1. 生成一个次级光源. 该步骤与 RSM 相同, 即对每个光源生成 SM.

Step2. 将虚拟光源注入到体素中. 体素中可能有多个 VPL, LPV 简化地视每个体素仅有一个位于中心的 VPL, 其效果应当等于位于该体素中的所有 VPL 叠加的结果. 为了进行上述叠加, 引入二阶球谐函数(spherical harmonics, SH)描述它在球面各个方向上的辐射度分布, 取得了良好的效果; 此外, 在叠加之前先将 VPL 位置沿法线方向移动半个体素的长度, 再注入移动后所在的体素, 以缓解自光照与自阴影问题.

Step3. 用一个迭代的过程传播光照. 在每次迭代中, 体素将自己存储的光照信息传递给相邻的 6 个体素, 计算目标体素各个面的入射光照, 并以此更新目标体素中心的 VPL 的出射光照. 光照从当前体素传递到相邻体素时, 需要计算相邻体素中另外 5 个面所接收到的光照的情况. 由于使用低阶 SH 估计这一光照是非常不准确的, 因此 LPV 算法计算当前体素中心到目标面的立体角大小, 使用其中心方向的光通量作为平均光通量.

上述计算得到的目标体素表面的光通量应等于目标体素中心 VPL 的发射光通量. 类似注入阶段, 算法更新目标体素中心 VPL 的 SH 系数, 用于下一次迭代. 经过 4~5 次迭代传播, 就能够得到较为稳定的结果; 使用着色点所在体素各个方向的入射辐射度, 即可解出渲染方程积分, 并进行渲染.

LPV 能够实时地模拟较真实的光照传播效果. LPV 的一个问题是体素划分的大小问题. Kaplanyan 等[12]基于细节层次(levels of detail, LoD)借助级联机制实现自适应的体素分割; 此外, 针对该算法和 RSM 一样在传播过程中忽略了遮挡和可见性的判断的问题, 提出一种场景的粗略表示, 用于取消应当被遮挡的位置的光传播, 表示为一个额外的球谐网格, 其位置和方向则根据实际场景而定; Franke[13]扩展了 LoD 方法, 并将其用于混合现实.

LPV 视每个体素的虚拟点光源都在体素中心, 因而无法区分物体在同一体素内的位置, 导致漏光情况; 如果体素的划分过于稀疏, 会加剧该问题.

2.1.3 VXGI 算法

由于 RSM 使用 SM 中的纹素作为次级光源,

因此难以高效地进行查询和访问. 而将场景划分为使用树状结构管理的体素, 以体素作为次级光源则能够更好地解决这一问题.

Crassin 等[14]提出的 VXGI 同样是实时全局光照方法. 渲染某个像素时, VXGI 访问体素树获取影响该像素对应场景点的体素, 通过体素中存储的光照信息完成渲染; 在渲染阶段, 在屏幕空间内执行锥形追踪. 该算法的思路是先将场景划分为体素, 再用 3D 数据结构管理它们, 分为体素化、存储体素信息和渲染 3 步.

Step1. 对场景进行体素化. 为了解决漏光问题, 将每个三角形都向周围扩展一些, 再离散化为体素, 并构建八叉树.

Step2. 存储体素信息. 与 LPV 记录各个方向的辐射度不同, 对于被光源直接照射的体素, VXGI 使用多元高斯分布函数记录直接光照的入射方向和位于体素内部的表面的法线分布.

Step3. 渲染. 对于每个像素对应的场景点, 考虑该点材质进行锥形追踪. 对于完美镜面反射材质, 计算反射光时仅需要计算出对应的入射方向; 对于光滑材质, 反射光需要考虑的入射方向可以简化为一个锥形, 该锥形利用树状结构获取与其相交的体素并计算光照贡献, 且在远处使用更大的体素以减少计算量; 对于漫反射表面, 则使用多个锥体来近似漫反射表面对光线的反射行为.

Panteleev[15]使用体素体积的 3D 裁剪纹理对 VXGI 加以改进; Aherne 等[16]则使用硬件加速的稀疏纹理计算间接光照; Franke[17]将锥形追踪扩展到混合现实环境中使用. 针对 VXGI 需要存储整个体素的反射辐射度信息的缺点, Sugihara 等[18]使用具有二进制信息的体素进行遮挡, 并使用预过滤的分层 RSM 进行间接光照计算; Chen 等[19]使用的方法与文献[18]类似, 但在紧凑列表中存储每个体素的辐射度.

与 LPV 算法相比, 由于需要维护 3D 的体素格子, 因此 VXGI 消耗的资源更多, 但质量也更好. 此外, 虚幻引擎 4 采用稀疏体素八叉树全局光照(sparse voxel octree global illumination, SVOGI), 使用稀疏体素八叉树管理体素, 更节约资源.

2.2　屏幕空间全局光照

屏幕空间全局光照算法指使用的信息仅为计算全局光照之前从屏幕上能够看到的信息, 即直接光照信息, 是对全局光照的近似, 是一种类似于后处理的手段.

2.2.1　屏幕空间环境光遮蔽

屏幕空间环境光遮蔽(screen space ambient occlusion, SSAO)是一种简单而容易实现的屏幕空间全局光照算法[20-22]. SSAO 算法基于以下几个假设: (1) 场景点接收到的各个方向的间接光照均为常量, 即环境光照; (2) 场景点某些方向无法接收到环境光; (3) 所有表面都是漫反射表面. 其理论依据为对渲染方程的估计, 即

$$L_{\mathrm{o}}(p,\omega_{\mathrm{o}}) \approx \frac{\int_{H^2} V(p,\omega_{\mathrm{i}})\cos\theta_{\mathrm{i}}\mathrm{d}\omega_{\mathrm{i}}}{\int_{\Omega+}\cos\theta_{\mathrm{i}}\mathrm{d}\omega_{\mathrm{i}}}\times \int_{H^2} L_{\mathrm{i}}(p,\omega_{\mathrm{i}}) f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})\cos\theta_{\mathrm{i}}\mathrm{d}\omega_{\mathrm{i}}.$$

其中, $L_{\mathrm{i}}(p,\omega_{\mathrm{i}})$ 表示各方向的入射光, 由上述假设, 计算间接光照时为常数; $f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})$ 表示 BRDF 项, 由于假设所有表面都是漫反射表面, 该项即为漫反射率. 因此, 最关键的问题就是如何确定 p 与间接光照的可见项 $V(p,\omega_{\mathrm{i}})$. SSAO 算法中间接光照由环境提供, 因此不考虑被其他物体遮挡的方向.

SSAO 算法使用屏幕空间场景的深度缓冲区而非真实的几何体数据来确定 $V(p,\omega_{\mathrm{i}})$. 如果法线贴图已知, 对于需要计算的场景点, 随机采样以其位置为球心、法线为中轴线的半球内的点, 判断这些点是否被其他像素遮蔽; 如果法线贴图未知, 则对整个球而非半球进行采样, 并且只有当被遮蔽的采样点超过总共采样点的一半的情况下, 才考虑进行环境光遮蔽问题.

针对采样步骤, Bavoil[23]提出水平基准环境光遮蔽(horizon-based ambient occlusion, HBAO)对各个采样方向进行加权计算. 与 SSAO 相比, HBAO 的采样率更低, 因此消耗硬件资源更少, 使得该算法对移动端更加友好, 在移动端上被广泛使用, 同时可以部分修正 SSAO 导致的遮挡错误问题.

SSAO 的显著优点是实现非常简单, 而且效果不错, 但由于采样方法较为简单, 在有几何中空等结构时可能对某些方向的可见性做出错误判断; 另外, 其对环境光各个方向入射光强度均相同的假设较为粗糙, 会出现一定的失真.

2.2.2　屏幕空间方向性遮蔽

Ritschel 等[24]对 SSAO 加以改进, 提出的屏幕空间方向性遮蔽(screen space directional occlusion, SSDO)算法, 指出像素对应的场景点本身即为次级光源, 间接光照由次级光源提供, 而非由环境提供, 这是其与 SSAO 的关键差异. 此时, 如果仍然将 $V(p,\omega_{\mathrm{i}})$ 定义为 p 与环境的可见项, SSDO 则与 SSAO 恰好相反. SSAO 中, 仅考虑可见项 V=1 的

方向; 而 SSDO 中, 仅仅考虑 V=0 的方向

$$L_{\mathrm{o}}^{\mathrm{indir}}\left(p,\omega_{\mathrm{o}}\right)=\int_{H^2,V=0}L_{\mathrm{i}}^{\mathrm{indir}}\left(p,\omega_{\mathrm{o}}\right)f_{\mathrm{r}}\left(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}}\right)\cos\theta_{\mathrm{i}}d\omega_{\mathrm{i}}.$$

SSDO 同样使用半球采样判断可见性, 如果采样点被遮挡, 则计算间接光照贡献.

通过使用屏幕空间中像素对应的场景点作为次级光源, SSDO 能够模拟出物体之间的相互光照, 增强全局光照的渲染质量, 达到比 SSAO 更真实的渲染效果; 但该算法仍然只能计算半球内的光源贡献, 无法考虑整个屏幕空间的信息.

2.2.3 屏幕空间反射

McGuire 等[25]提出的屏幕空间反射(screen space reflection, SSR)算法是一种在屏幕空间上模拟光线追踪的算法. 在 SSR 算法出现之前, 已有一些通过在屏幕空间上模拟光线追踪计算间接光照的方法. 由于直接追踪每根光线所需的计算量太大, 研究人员提出了几种减少计算量的方法. Sloan 等[26]利用屏幕空间信息生成一组 SH 来计算间接光照; Nichols 等[27-28]提出通过分割几何缓冲区的方法来提取一组 VPL, 并将 VPL 聚集成簇, 从下到上分层, 加速间接光照计算. SSR 算法则更加充分地利用屏幕空间信息, 对于每个像素, 其与光线追踪类似, 使用蒙特卡罗积分法求解渲染方程. 为了减少计算量, 加速光线采样, SSR 使用数字微分法(digital differential analyzer, DDA)在屏幕空间上作光线步进, 并使用深度贴图的多级渐远纹理优化步进的计算开销.

当前, SSR 算法已经能够渲染出相当真实的间接光照. 如果在时间上进行样本累积, 仅考虑屏幕信息的情况下, 已经能够趋近于无限次弹射的间接光照.

2.3 对比与分析

对于光栅化全局光照, 支持的光源类型不如光线追踪复杂. 如 RSM 需要 SM 信息, 而复杂面光源难以渲染对应 SM, 因此 RSM 多用于点光源和方向光源等简单光源场景.

本节介绍的光栅化全局光照代表性算法中, 其计算代价、间接光照效果、材质要求及部分改进方法的对比如表 1 所示. 其中, 由于深度/几何缓冲区在光栅化渲染流水线中可以认为是每帧都需要渲染的, 因此获取代价极小, 计算代价列中考虑这二者之外的额外计算代价; SSAO 与 SSDO 一般仅使用直接光照渲染出的颜色缓冲区, 但也可以通过几何缓冲区获取 BRDF 进行计算, 并不要求漫反射材质, 因此将这 2 种方法记为材质均支持.

屏幕空间算法的固有问题是未考虑超出屏幕范围之外或者被遮挡的间接光照. 在实际应用中, 这一问题往往通过与其他方法结合得到解决. 虽然 3D 空间全局光照算法的计算代价更高, 但由于能够考虑屏幕外的信息, 因此能获得更加精确的一层间接光照效果.

表 1 光栅化全局光照代表性算法对比

算法	计算代价	间接光照效果	材质要求	部分改进方法
RSM[5]	RSM 及采样	一层间接光照的粗略估计, 忽略遮挡	仅漫反射	计算代价[6-7], 近似估计遮挡[8-9]
LPV[7]	体素划分, RSM, 辐射度注入与传播	一层间接光照的较好估计, 忽略遮挡	仅漫反射	体素划分[12-13]
VXGI[14]	层次体素划分, RSM, 锥形追踪	一层间接光照	支持光滑面	计算代价[15-16], 体素辐射度存储[18-19]
SSAO[20-22]	半球采样	认为环境光强度一致且仅考虑环境光	均支持	HBAO[23]
SSDO[24]	半球采样	用屏幕像素对应的场景点估计间接光照, 模拟物体相互光照	均支持	
SSR[25]	屏幕空间 DDA 光线步进	充分使用屏幕空间信息完成间接光照, 可以趋近真实	均支持	

3 光线追踪实时全局光照

光线追踪算法是渲染的重要手段, 能支持绝大部分复杂场景和光源. 基于光线追踪时是否用到专门硬件单元, 可分为硬件光线追踪和软件光线追踪. 通常, 调用 GPU 中用于加速光线追踪求交的专门硬件逻辑电路单元即称为硬件光线追踪, 这些硬件单元包括 NVIDIA RTX 系列 GPU 的 RT Core, AMD GPU 的 Ray Accelerators 等. 不调用上述专用硬件单元, 仅依赖 CPU 或 GPU 的通用计算能力进行计算, 即为软件光线追踪.

软件和硬件光线追踪都有对应的光线求交加速手段, 以满足实时渲染的实时性要求, 使得光线追踪在实时渲染中逐渐得到应用, 并且成为研究热点. 经过上述求交加速后, 通常, 每帧采样的光线数量是每像素样本数(sample per pixel, SPP), 约

为每个像素采样一根光线, 即 1 SPP. 然而这意味着在蒙特卡罗估计中采样数 $N=1$. 因此, 如果仅仅使用均匀球面采样进行蒙特卡罗估计, 会导致方差较大, 进而形成过暗或者过亮的像素, 即产生明显噪点. 解决噪点问题的方法包括:

(1) 更合理的采样方案. 优化蒙特卡罗估计式的采样分布, 让其更接近渲染方程积分的被积函数 $L_{\mathrm{i}}(p,\omega_{\mathrm{i}})f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})(\boldsymbol{n}\cdot\omega_{\mathrm{i}})$, 从而减少采样方差.

(2) 降噪. 基本手段是空间滤波与时间滤波. 前者通常对当前帧图像应用滤波核; 后者则通过计算运动矢量描述上一帧与当前帧像素的对应关系, 之后进行滤波.

(3) 神经网络参与的实时光线追踪. 近年来, 随着 GPU 性能的提升, 能够更快速地执行神经网络的训练与推断. 限于实时性要求, 大型神经网络在实时渲染中难以得到应用, 而小型神经网络正在逐渐取代传统降噪器, 并且成为研究热点. 使用神经网络可以构造像素信息或光线信息, 从而更有效地利用已有光线信息达到更好的渲染精度.

本节将依次讨论这些光线追踪加速方法, 其中, 求交加速之外的方法对比如表 2 所示.

表 2　光线追踪优化方法对比

优化方法	优点	缺点
采样方案	大多数无偏, 额外计算代价通常不高	对于复杂场景难以找到最优策略
降噪	实现简单, 效果好	一定计算开销, 难以处理动态复杂场景
神经网络	效果好	计算开销更高, 需要训练网络, 依赖训练数据, 硬件要求较高

3.1　软件光线追踪与硬件光线追踪

在实时渲染中, 渲染的实时性要求(≥30 帧/s)是所有实时渲染算法都需要考虑的. 对于光线追踪算法, 求交算法是时间瓶颈. 软件光线追踪常用的加速手段包括构建包围体层次结构(bounding volume hierarchies, BVH), 有向距离场(signed distance field, SDF)等. 而硬件光线追踪则通过 NVIDIA 等显卡厂商提供的高速硬件接口处理光线求交, 满足实时性.

BVH 是一种经典的加速手段, 但在场景面片数量极大的情况下, 使用软件实现难以满足实时性, 在实时应用中一般需要使用硬件逻辑电路, 即专门的硬件单元进行加速.

SDF 是一种经典的 3D 数据表示方法[29], 其定义了空间中任意一个点到物体边缘的最近距离. Evans[30]将 SDF 引入光线追踪算法中, 用于对光线追踪中的求交算法进行加速. 在光线追踪的过程中, 每次步进都将步进的距离设置为当前位置的 SDF 函数值, 保证步进时不与其他物体产生相交, 虚幻引擎[31]就使用了这种技术, 如图 2 所示. SDF 使得软件光线追踪得以大大加速, 并且在面片严重重合的复杂场景中速度更快.

图 2　使用 SDF 进行光线追踪[31]

在虚幻引擎 5 的实时全局光照技术报告中, Wright 等[32]对 Lumen 使用的光线追踪技术进行说明, 可以代表产业界光线追踪实现的前沿水平. 在 Lumen 的软件光线追踪技术中, 使用网格 SDF 和全局 SDF 分别负责近处与远处的光线追踪; 同时, 提出表面缓存, 对每个网格体生成投影板, 将网格体的材质信息, 即漫反射率、法线、发射率等存储至投影板纹理中, 供 SDF 查询交点信息时使用. Lumen 的硬件光线追踪则直接调用 GPU 提供的光线追踪接口.

Lumen 的光线追踪用于最终聚合和反射 2 个步骤. 在不同场景下, 对于上述 2 个步骤, Lumen 中软件与硬件光线追踪的性能对比如表 3 所示.

表 3　Lumen[32]中软件/硬件光线追踪性能对比 ms

测试场景	软件光线追踪	硬件光线追踪
最终聚合: Lumen in the Land of Nanite	0.94	10.03
反射: Lumen in the Land of Nanite	0.04	0.46
最终聚合: Lyra	1.21	1.41
反射: Lyra	1.58	2.03
最终聚合: The Matrix Awakens	1.83	2.13
反射: The Matrix Awakens	1.76	2.52

综上所述, 软件光线追踪的优点是其灵活性, 它能够在各种操作系统平台上运行, 不需要显卡中有光线追踪单元. 如果使用 SDF 进行加速, 能够在舍弃部分精确性的前提下, 取得与硬件光线

追踪相似甚至更快的速度. 但是, 其限制是难以支持部分几何类型, 如蒙皮网格. 硬件光线追踪则是由显卡提供的硬件接口, 场景加速结构由显卡进行管理, 能够提供更加准确的追踪结果, 同时可以支持所有几何类型.

3.2 光线追踪采样方法

光线追踪是为了求解渲染方程积分. 为了简便, 将被积函数记作 $f(x)\equiv L_{\mathrm{i}}(p,x)f_{\mathrm{r}}(p,x,\omega_{\mathrm{o}})(\boldsymbol{n}\cdot x)$. 重要性采样估计式可以重写为

$$L\approx\langle L\rangle_{is}^{N}=\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}.$$

当 $f(x_i)$ 与 $p(x_i)$ 成正比时, 取到零方差值, 即最小方差. 然而, 基于光线在场景中传播的复杂性, 直接基于 $f(x)$ 进行采样是难以实现的. 因此, 采样策略的优化分别基于被积函数的 3 个因式进行考虑: (1) 光源照射场景, 场景中的光线辐射度分布; (2) p 的 BRDF; (3) 光线方向与 p 法线夹角的余弦值; 它们对应若干种采样策略. 多重重要性采样(multiple importance sampling, MIS)[28]可以将 M 种采样策略结合, 其中, 第 s 种采样策略的采样样本数为 N_s. 则 MIS 估计量为

$$\langle L\rangle_{\mathrm{MIS}}^{M,N}=\sum_{s=1}^{M}\sum_{j=1}^{N_s}\frac{1}{N_s}w_s(x_{ij})\frac{f(x_{ij})}{p_s(x_{ij})}.$$

其中, w_s 表示第 s 种采样策略的 MIS 权重, MIS 估计量无偏, 需要满足 M 种采样策略的权重之和为 1, 且对任意样本空间上的 x 均成立. 一种常用的权重为平衡启发权重[33-34], 即

$$w_s=N_s p_s(x)\Big/\sum_j N_j p_j(x).$$

基于 BRDF 和余弦项采样均有成熟的采样方案, 并已经得到广泛应用. $L_{\mathrm{i}}(p,x)$ 则是最难以估计的一项因式, 路径引导算法[35-38]虽然能够在光线追踪的过程中学习采样分布, 但均难以满足实时性要求.

2020 年, Bitterli 等[39]将加权蓄水池采样(weighted reservoir sampling, WRS)和重采样重要性采样(resampled importance sampling, RIS)[40]应用于光线追踪算法中, 提出时空蓄水池重要性重采样(reservoir spatio-temporal importance resampling, ReSTIR). 该算法在时间上共享相邻帧的像素信息, 在空间上共享相邻像素信息, 从而在多动态光源情况下有效率地进行直接光照(direct illumination, DI)估计, 因此也被称为 ReSTIR DI, 其有效样本对应的采样分布能够更加接近 $f(x)$, 从而取得更好的渲染效果.

在 ReSTIR DI 的基础上, Ouyang 等[41]完善了对应的全局光照算法 ReSTIR GI, 在支持硬件光线追踪的设备上能够达到实时要求, 且效果明显优于 1 SPP 的路径追踪结果, 如图 3 所示. Lin 等[42]提出 ReSTIR PT, 将算法拓展至离线场景下的路径追踪.

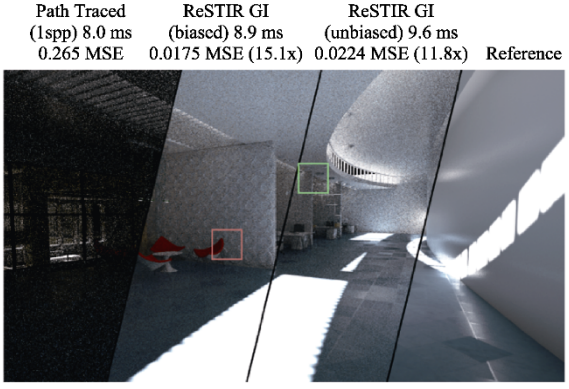

图 3 ReSTIR GI 与路径追踪对比[41]

3.3 实时光线追踪降噪

实时光线追踪的降噪问题即为对蒙特卡罗渲染的实时降噪问题, 由于其具有实时性的要求, 蒙特卡罗渲染采样的光线数通常较小, 可以分为空间域和时间域的降噪, 二者通常在实时渲染过程中共同应用. 2009 年, Robison 等[43]首次对单幅 1 SPP 图像进行降噪. 下面将分别讨论空间域和时间域的传统降噪算法, 并介绍基于神经网络的实时降噪.

空间域进行降噪的常用手段是滤波, 其不考虑相邻帧提供的信息, 只考虑当前帧的渲染信息. 可以分为 2 类方法:

(1) 将实时渲染中的直接光照和间接光照作拆分, 并且对后者进行滤波, 包括使用非抽样小波变换进行滤波[44-45], 使用自适应高维滤波核作实时高维滤波[46], 分析图像中的边缘信息指导滤波过程[47]等. 与离线降噪手段相比, 由于此类手段不采用误差估计, 因此可能造成局部细节的损失[48].

(2) 基于传输过程中的频域信息进行滤波. Mehta 等[49]使用轴对齐滤波器对软阴影、漫反射间接光照[50]进行滤波; Yan 等[51]则通过拆分滤波核对四维错切滤波核进行加速. 这类方法在常用场景下能达到更快的速度, 但准确性相对更差; 同时, 其用时与场景中独立面光源的个数相关, 难以扩展到规模更大的场景中.

时间域降噪则通过利用多帧的信息, 能够在

低采样率的情况下提高渲染的稳定性. 在离线场景下, 时间域降噪的输入为一段视频, 可以看作一系列预计算完成的帧. 此时, 在时间域滤波可以使用更多的数据, 包括当前帧之前及之后的帧. Meyer 等[52]对所有帧作主成分分析(principal component analysis, PCA)并丢弃不显著的基向量, 以提高时域稳定性; Delbracio 等[53]则通过考虑 3 帧的光线直方图缓解闪烁问题.

然而在实时渲染场景中, 因为当前帧之后的帧还未完成渲染, 所以当前帧无法使用所有帧的信息. 实时时域滤波方法多为基于光流的算法, 即通过使用运动矢量对一帧的像素重投影到另一帧[54-55]. 2014 年, Karis[56]受重投影的思想启发, 提出时域反走样(temporal anti-aliasing, TAA), 并且得到了广泛的应用.

2017 年, Schied 等[6]提出方差指导的时空滤波(spatiotemporal variance-guided filtering, SVGF), 使用时域信息增加有效样本数并且估计方差, 以驱动图像空间的小波滤波器. 在对分辨率为 1 920×1 080 的 1 SPP 的路径追踪结果进行降噪重建时, SVGF 的重建耗时约为 10 ms, 渲染效果能够接近 2 048 SPP 的离线路径追踪渲染结果, 如图 4 所示.

图 4　1 SPP 路径追踪结果(左), SVGF 重建后结果(中), 2 048 SPP 路径追踪图(右)[6]

3.4　神经网络参与的实时光线追踪

随着 GPU 硬件算力的增长, 深度学习在实时渲染中逐渐得到应用. 深度学习能够构造像素信息, 通过超分辨率减少光线追踪所需渲染的像素数量; 或直接构造光线追踪所需的信息, 以减少计算量, 完成对实时光线追踪的进一步加速.

2017 年, Chaitanya 等[57]使用递归自编码器的深度神经网络(deep neural network, DNN)完成低采样率的可交互式渲染降噪, 但是耗时超过 50 ms, 难以满足实时性需求.

2018 年, NVIDIA 推出深度学习超分辨率(deep learning super sampling, DLSS)[58]; 2020 年, 推出 DLSS2.0[59]; 2022 年, 推出 DLSS3.0. DLSS 将深度学习引入实时全局光照渲染, 能够替代传统降噪手段; 充分使用图灵架构 GPU 的张量计算核心, 实现了 DNN; 使用预训练网络, 不需要在用户端进行实时训练; 使用卷积自编码层处理渲染所得的数据, 以获得高维特征, 再使用解码器将高维特征转化为颜色信息. 与传统降噪方式类似, DLSS 超采样同样体现在空间域和时间域 2 个维度: 在空间域上, 将低分辨率渲染图处理为高分辨率图像, 减少了高分辨率图像中每个像素需要追踪的光线数量; 在时间域上, 使用神经网络完成帧生成, 减少了单位时间需要使用渲染管线生成的帧数. 这些手段都有效地加速了实时光线追踪.

之后, AMD 与 Intel 这 2 家硬件厂商相继推出 AMD FSR[60]和 Intel XeSS[61], 它们均使用 GPU 完成了对超分辨率技术的支持.

超分辨率技术在学术界也有相关的研究工作. 2021 年, Hadadan 等[62]提出 Neural radiosity 方法, 用深度学习的神经网络替代传统的辐射度计算模型. 与传统方法依赖预定义的基函数不同, 该方法通过神经网络直接学习四维辐射度分布, 以满足实时渲染的需求; 同时, 优化了渲染方程的残差范数, 使其适应包括漫反射和非漫反射在内的多种需求. 2023 年, Zheng 等[63]提出一种模块化的神经渲染方法 NeLT, 通过为每个对象学习其光传输特性, 实现对动态场景的高效渲染; 将复杂的全局光照过程分解为多个对象的光传输函数, 并在不同对象之间显式地组合这些函数, 实现灵活的动态光照效果. 得益于这种分解方法, NeLT 能够实现动态阴影、漫反射和镜面反射等复杂光照效果, 且渲染质量与先进的去噪方法相当, 在动态场景中具备良好的时间稳定性和多视图一致性. 2023 年, Wu 等[64]提出 ExtraSS, 将空间域超采样与时间域帧外推结合到一个框架中, 能够准确地处理移动阴影, 并且生成在时域上稳定的结果. DLSS 使用帧插值完成时域超采样, ExtraSS 使用的帧外推能够进一步减少渲染延迟. 帧外推的研究还包括使用光流进行帧外推[65], 使用遮蔽动作矢量和几何缓冲区信息进行帧外推[66], 但是在复杂场景和遮挡关系有变化的区域中存在错误. ExtraSS 解决了上述问题, 同时需要输入的像素数量更少. Zhong 等[67]提出 FuseSR, 以低分辨率图像和高分辨率几何缓冲区作为输入, 提高了空间域超分辨率的性能. Zhang 等[68]引入高频傅里叶映射和低频残差学习模块, 完成对渲染结果图的任意倍率超采样.

除超分辨率外, 神经网络也被用于其他方面以达成减少光线追踪计算量的效果. 2021 年, Müller 等[69]提出神经辐射度缓存(neural radiance cache, NRC), 使用神经网络实时生成动态场景的辐射度缓存, 每帧更新与查询辐射度缓存仅需 2.6 ms, 但会引入偏差.

4 探针法实时全局光照

探针法全局光照中一般使用光线追踪计算辐照度函数并进行缓存, 在渲染时则使用光栅化, 以及已经计算好的探针辐照度插值估计像素的颜色. 因此, 可以看作一种结合了光栅化的快速和光线追踪的准确性的混合方法. 本质上, 探针是一种缓存的手段, 将用于渲染的局部信息保存至探针纹理, 在渲染时可以直接调用, 以减少计算量.

4.1 辐照度探针法

对于漫反射表面(或粗糙表面), 其 BRDF 是一个常数, 即与光线的入射和出射方向无关. 给定漫反射率, 可以将原渲染方程中的积分化简为

$$\int_{\Omega+} L_{\mathrm{i}}(p,\omega_{\mathrm{i}}) f_{\mathrm{r}}(p,\omega_{\mathrm{i}},\omega_{\mathrm{o}})(\boldsymbol{n}\cdot\omega_{\mathrm{i}})\,\mathrm{d}\omega_{\mathrm{i}} = \frac{\rho}{\pi}\int_{\Omega+} L_{\mathrm{i}}(p,\omega_{\mathrm{i}})(\boldsymbol{n}\cdot\omega_{\mathrm{i}})\,\mathrm{d}\omega_{\mathrm{i}} = \frac{\rho}{\pi}E(p).$$

其中, $E(p)$为 p 接收到的辐照度, 描述该点从空间中各个方向接收到的光线能量之和. 在这种情况下, 渲染方程关注的问题转化为某个漫反射点 p 从轴线为其法线 $\boldsymbol{n}$的半球方向接收到的辐照度. Greger 等[70]提出辐照度体的概念, 对空间中的任一位置 p, 均可假设存在法线为 $\boldsymbol{n}$的微表面, 从而计算辐照度 $E(p,\boldsymbol{n})$, 即辐照度分布函数, 可以用辐照度体进行离散化存储, 这就是辐照度探针法的理论基础.

基于此, McGuire 等[71]提出预计算光场探针法, 将一个长方体区域划分为探针体, 探针体沿坐标轴方向均匀地划分为小长方体体素, 探针位于每个体素中心. 其思路是使用探针体中的探针保存周围的物体与探针的距离, 以及探针接收到的辐照度分布函数信息, 计算场景点着色时, 使用点周围的探针保存的对应方向的辐照度进行三线性插值; 由于这种方法基于预计算, 因此难以支持动态场景的全局光照.

2019 年, Majercik 等[72]提出动态漫反射全局光照(dynamic diffuse global illumination, DDGI), 使用八面体映射维护探针对应方向的距离和辐照度信息, 这二者均为假想微表面(位置为探针坐标 p, 法线为 $\boldsymbol{n}$)的信息, 分别描述了 p 处向着 $\boldsymbol{n}$方向最近的物体距离以及该微表面接收到的辐照度. 本质上, 这是对距离函数和辐照度分布函数的离散化存储; 然后将离散化的球面方向映射到以球心为中心的八面体纹素, 再将八面体纹素映射到 2D 纹素进行保存. 为了支持动态场景, DDGI 使用光线追踪对上述纹理进行更新, 每帧均以探针为起点追踪光线, 将光线追踪的结果用于计算纹理数据. 对于任意一个位于探针体内部的、法线为 $\boldsymbol{n}$的点 p, 它总是位于某 8 个探针围成的长方体探针笼(类似于长方体体素)内部. 由于辐照度在空间中的分布是较为平滑的[73], 因此计算 p 的辐照度时, DDGI 获取这 8 个探针处法线为 $\boldsymbol{n}$的微表面辐照度信息, 三线性插值估计 p 处的辐照度, 且上述插值使用探针纹理中的距离信息进行切比雪夫遮挡测试, 估计探针与 p 之间的可见性.

之后, Majercik 等[74]对 DDGI 进行优化, 开源了 NVIDIA RTXGI 1.0; 同时, 对非漫反射材质的表面光照提出一种估计方式, 使得 DDGI 算法能够在虚幻引擎等渲染引擎中得到应用. RTXGI 中, 增加了对探针是否活跃的区分和对探针的偏移, 并且实现了随相机移动的探针体. 对于镜面反射材质, 使用反射方向微表面的辐照度估计反射方向的入射辐射度. Hu 等[75]则使用 SDF 软件光线追踪替代 DDGI 中使用的硬件光线追踪.

Majercik 等[76]将 DDGI 与 ReSTIR DI[39]算法结合到一个统一的框架中, 将 DDGI 查询延迟至次级路径顶点, 使 DDGI 成为场景的光源, 然后利用 ReSTIR 的重采样机制统一了直接和间接光照的采样. 通过将 2 种技术进行有机结合, 实现了低噪声和更少偏差的全局光照估计, 有效地支持实时应用.

与光线追踪实时光照直接从像素追踪光线再加以降噪相比, DDGI 类辐照度探针算法使用探针体保存离散化的辐照度场, 能够减少计算量, 同时也是对于漫反射表面光照的良好近似. 然而, DDGI 的不足之处在于:

(1) 对于光滑表面, 使用辐照度计算渲染方程积分在理论上是有缺陷的, 因为其 BRDF 与漫反射表面不同, 并不是一个常值函数.

(2) 辐照度并不是真正在像素对应位置计算得到的, 以导致漏光和过遮挡的情况.

(3) 自动摆放的探针体会导致部分探针位于物体内部或位于场景外, 加重漏光或过遮挡问题.

基于面元的全局光照[77]能够处理上述探针摆放问题, 本文将其也归于辐照度探针法一类. 这是因为面元可以看作一种放置于几何表面的探针, 其中存储位置、法线、面元半径、辐照度缓存等信息可以看作辐照度探针的变种; 能够动态地更新探针位置, 与辐照度探针一样向场景中追踪光线以计算间接光照; 使用面元还可以优化光线采样方向.

4.2　辐射度探针法

辐射度探针法则是对辐照度探针法与光线追踪法的折中, 能够在保证一定渲染准确性的前提下减少光线追踪计算开销.

Lumen[32]在其最终聚合阶段, 提出了屏幕探针和世界空间辐射度探针.

屏幕探针是基于屏幕空间自适应摆放的探针, 本质上是更稀疏地对像素进行光线追踪着色. 为了完成探针的摆放, Lumen 使用自适应的降采样策略, 即在更细致的几何体处摆放更多探针. 首先在屏幕空间中均匀摆放探针, 在每个方向约为 1/16 分辨率, 即在每个方向上隔 16 个像素放置一个探针, 其位置可以使用屏幕坐标以及对应的深度缓冲区中的深度信息, 将其映射到 3D 世界坐标, 实际上也就是探针所在像素对应的场景点位置; 然后对每个探针周围的像素进行插值测试, 如果插值失败, 那么在 2 个探针上直接额外摆放一个探针, 并重复该过程. 这个过程将在细分层级深度到达预设限制、失败像素数量到达阈值等条件下终止. 屏幕探针纹理同样存储距离信息, 但与辐照度探针不同, 屏幕探针存储对应方向的辐射度信息. Lumen 考虑 BRDF 等信息, 对屏幕探针使用重要性采样进行光线追踪, 同时对探针纹理使用时间域和空间域滤波以减少噪声.

世界空间辐射度探针的放置类似于 DDGI[72], 在一个更稀疏的尺度上放置探针, 但每个探针追踪的光线数量更多, 当屏幕探针无法追踪到某些光源时, 利用世界空间辐射度探针可以处理离屏幕更远的光照, 以弥补屏幕辐射度探针追踪光线数量较少的手段. 如一个离屏幕很远的小窗口, 这是一个很难被屏幕探针追踪到的光源.

2023 年, Boissé 等[78]提出 GI-1.0, 使用动作矢量评估上一帧的探针与当前帧像素的对应关系, 并且将上一帧对应的探针纹理直接填充到当前帧的探针纹理中, 进一步减少需要更新的屏幕探针数量, 达到减少更新探针计算量的效果. 为了更好地捕获场景中的动态信息, GI-1.0 中每帧强制生成 1/4 的探针, 并且使用 3 种队列决定最终生成的探针位置; 对同一像素块使用最近最少使用(least recently used, LRU)缓存, 对该像素块的探针信息进行换入与换出, 虽然可以在一定程度上缓解因屏幕探针采样率低导致的闪烁现象, 但会增加若干倍探针存储开销.

4.3　对比与分析

探针法本质上是一种缓存手段, 其关键区别在于如何摆放探针、探针中存储的信息、如何更新探针信息, 以及如何使用探针信息进行渲染. 其中, 摆放探针的方式包括在世界空间摆放、在屏幕空间摆放、在物体表面摆放等. 辐照度探针和辐射度探针的区别在于离散化存储的对象是辐照度分布函数还是辐射度分布函数, 其中, 辐照度探针的难题是解决目前仅能处理漫反射间接光照的局限性, 而辐射度探针的难题则更多是计算开销的问题. 更新探针信息是支持动态全局光照的必要步骤, 暂时仅有通过光线追踪进行探针更新一种方法. 随着神经网络逐渐应用于实时渲染, 本文认为, 通过神经网络更新部分探针纹理、完成探针渲染的插值等, 也将成为新的研究方向. 实际上, 对上述部分进行适当搭配, 即可组合出一种对应的研究方向. 因此, 探针法全局光照也是实时渲染的重要研究方向.

5　代表性算法对比

无预处理的实时全局光照方法均支持动态场景, 各类算法对比如表 4 所示.

从表 4 可以看出:

(1) 光栅化实时全局光照对硬件的要求低, 渲染速度快, 是应用最广泛的实时全局光照渲染技术, 也是研究最充分的实时全局光照技术. 根据是否仅使用全局光照渲染出的屏幕信息, 光栅化实时全局光照分为 3D 空间全局光照和屏幕空间全局光照. 前者能够使用更多信息计算更精确的间接光照, 但时间与空间开销相对更大; 后者由于仅使用屏幕信息, 因此存在一些错误估计的情况.

(2) 光线追踪实时全局光照本质上使用蒙特卡罗法解渲染方程积分, 因此能够渲染更真实的光照效果, 但对硬件算力要求更高. 软件光线追踪使用 SDF 进行求交判断, 硬件光线追踪则进一步要求 GPU 硬件支持光线追踪接口. 由于实时性的要求, 此类方法仅能支持约 1 SPP, 因此需要额外的降噪手段, 以减少蒙特卡罗估计的方差导致的

表 4 无预处理的实时全局光照渲染代表性算法对比

算法类型	算法名称	硬件要求	材质要求	渲染效果
3D 空间	RSM 类[5-9]	○○○	仅漫反射	不适应复杂光源, 不考虑遮挡, 一层间接光照粗略估计
	LPV 类[7,12-13]	○○○	仅漫反射	不考虑遮挡, 一层间接光照的较好近似
	VXGI 类[8,15-19]	○○○	支持光滑面	一层间接光照
屏幕空间	SSAO[20-22]	○	均支持	仅考虑环境光间接光照
	SSDO[24]	○	均支持	仅考虑屏幕像素间接光照
	SSR[25]	○○	均支持	屏幕空间算法中效果最好, 但无法考虑屏幕外的信息
光线追踪	硬件光线追踪	○○○○	均支持	搭配超分辨率技术能够快速且真实地完成渲染
	软件光线追踪	○○○	均支持	使用 SDF 加速能够极大地加速, 但较硬件光线追踪存在误差; 使用 BVH 则不如硬件光线追踪速度快
探针法	DDGI 类[72,74,76]	○○○	仅漫反射	较好地估计漫反射表面全局光照效果, 改进方法可以对光滑面作粗略估计
	Lumen: 屏幕[32]	○○○	均支持	效果较好, 计算代价略高于世界空间探针

注. ○表示硬件要求高低程度, ○数量越多, 硬件要求越高. ○: 计算量较小, 能够在移动端设备上运行; ○○○○: 计算量较大, 需要在支持光线追踪的显卡上运行.

屏幕噪点, 包括空间域和时间域上的降噪; 随着 GPU 性能的提高, 神经网络正逐渐被引入光线追踪流程, 其中, DLSS 技术已经在产业界应用.

(3) 探针法实时全局光照本质上是一种缓存光线追踪结果的手段, 通过将光线追踪的对象从像素变为稀疏化的探针, 可以对探针进行更多次光线追踪, 计算探针光照纹理后光栅化插值计算像素着色, 以减少计算开销. 根据探针纹理保存的信息, 可以分为辐照度探针法和辐射度探针法, 其中, 辐照度探针利用空间中辐照度分布变化较为平滑的性质[73], 能够较好地解决漫反射表面的间接光照问题; 辐射度探针则略微增加计算开销, 使用屏幕探针避免了探针摆放的问题, 可以保存光滑表面的辐射度, 比辐照度探针的精确性更高.

6 结 语

实时全局光照渲染是真实感渲染的重要研究方向. 本文对无预处理的实时全局光照渲染的主要方法进行综述, 将其分为对光栅化、光线追踪和探针法实时全局光照 3 大类进行分析对比, 并总结了各类方法的优缺点.

受限于硬件速度与场景复杂度, 本文综述的方法均有其对应的适用场景, 在成熟的渲染引擎中, 通常将以上方法结合使用, 并且针对不同硬件平台使用不同的渲染策略. 在硬件算力较差时, 如移动端, 使用光栅化等更快速的渲染方法; 而在硬件算力高、显卡能够支持光线追踪时, 则在保证实时性的前提下使用更精确的渲染方法. 本文认为, 随着硬件算力的提升, 未来实时光照渲染的研究将会向更精确的光照计算发展:

(1) 光栅化算法中, 屏幕空间全局光照算法已经相当成熟, 未来需要进一步结合额外的信息. 对于 3D 空间全局光照算法, 需要从速度和精确度 2 个方面进行改进, 可以在保证高速的情况下尝试引入轻量级的光线追踪和神经网络进行优化.

(2) 在离线渲染场景下, 光线追踪方法能够提供非常逼真的渲染效果, 然而其中一些方法, 如路径引导技术[35-38], 由于时间限制, 在实时渲染中的应用充满挑战. 2023 年, Dong 等[79]提出 NPM 方法, 通过神经隐式表示建模目标分布, 实现了更高效的路径引导, 为这种方法在实时全局光照管线中的应用提供了帮助. 如何进一步将这些方法进行适应于实时渲染的修改是一个研究方向. 此外, 在光线追踪方法中, 对光线追踪的求交过程以及渲染图降噪使用神经网络进行辅助, 也成为近年来的研究热点.

(3) 探针法作为光栅化与光线追踪的混合方法, 目前其中虽然有对探针的时间域和空间域滤波, 但是仍然缺乏对探针本身的光线追踪进行进一步优化, 如使用神经网络优化探针的光线追踪过程、探针的摆放、探针对像素的插值过程等, 或者使用 ReSTIR[39]等方法优化探针本身的光线追踪过程, 而非仅仅使用 ReSTIR 计算直接光照[76].

从本文提到的各种技术中可以总结出若干过去技术的发展趋势. 首先是硬件加速推动的性能提升, 以 NVIDIA 为代表的硬件的进步极大地推动了实时全局光照技术的发展; 其次是神经网络和深度学习技术的引入, 近年来, 将神经网络参与或取代渲染过程中的各个模块的做法越来越常见, 这

些技术在渲染质量、渲染效率上的优良表现都展现出神经网络在实时光照渲染中的巨大优势和潜力.

目前，仍存在不少亟待解决的问题：很多算法都在动态场景的处理中存在较大瓶颈，尤其是对象复杂移动、光源变化剧烈的情况；虽然众多高端 GPU 和硬件加速的实时渲染有优异的表现，但是高硬件依赖让这些方法难以得到普及；由于实时渲染中采样数量有限，噪点问题仍然存在，并且降噪过程通常会消耗大量的计算资源等.

参考文献(References):

[1] Sloan P P, Kautz J, Snyder J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments[J]. Seminal Graphics Papers: Pushing the Boundaries, 2023, 2: Article No.37

[2] Cohen M F, Greenberg D P, Immel D S, *et al*. An efficient radiosity approach for realistic image synthesis[J]. IEEE Computer Graphics and Applications, 1986, 6(3): 26-35

[3] Xu Xiang, Wu Xiaolong, Chen Ziling, *et al*. A survey of ray tracing rendering of large-scale scenes[J]. Journal of Computer-Aided Design & Computer Graphics, 2024, 36(8): 1155-1170 (in Chinese)
(徐翔, 吴小龙, 陈子凌, 等. 大规模三维场景光线追踪渲染方法综述[J]. 计算机辅助设计与图形学学报, 2024, 36(8): 1155-1170)

[4] Kajiya J T. The rendering equation[C] //Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 1986: 143-150

[5] Dachsbacher C, Stamminger M. Reflective shadow maps[C] //Proceedings of the Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2005: 203-231

[6] Schied C, Kaplanyan A, Wyman C, *et al*. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination[C] //Proceedings of High Performance Graphics. New York: ACM Press, 2017. Article No.2

[7] Kaplanyan A. Light propagation volumes in CryEngine 3[OL]. [2024-11-09]. https://www.semanticscholar.org/paper/Light-Propagation-Volumes-in-CryEngine-3-Kaplanyan/969ea62cbdb62a03c8fd0100b62a6f01fe5465b9

[8] Ritschel T, Grosch T, Kim M H, *et al*. Imperfect shadow maps for efficient computation of indirect illumination[J]. ACM Transactions on Graphics, 2008, 27(5): Article No.129

[9] Ritschel T, Eisemann E, Ha I, *et al*. Making imperfect shadow maps view-adaptive: high-quality global illumination in large dynamic scenes[J]. Computer Graphics Forum, 2011, 30(8): 2258-2269

[10] Wu Xiangyang, Chai Xueliang, Wang Yigang, *et al*. Form-factor sampling based real-time global illumination[J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(6): 941-948(in Chinese)
(吴向阳, 柴学梁, 王毅刚, 等. 利用形状因子采样的实时全局光照绘制[J]. 计算机辅助设计与图形学学报, 2011, 23(6): 941-948)

[11] Xu Xiang, Wang Lu, Xu Yanning, *et al*. A survey of point based global illumination[J]. Journal of Computer-Aided Design & Computer Graphics, 2019, 31(5): 689-697(in Chinese)
(徐翔, 王璐, 徐延宁, 等. 基于点的全局光照绘制方法综述[J]. 计算机辅助设计与图形学学报, 2019, 31(5): 689-697)

[12] Kaplanyan A, Dachsbacher C. Cascaded light propagation volumes for real-time indirect illumination[C] //Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2010: 99-107

[13] Franke T A. Delta light propagation volumes for mixed reality[C] //Proceedings of the IEEE International Symposium on Mixed and Augmented Reality. Los Alamitos: IEEE Computer Society Press, 2013: 125-132

[14] Crassin C, Neyret F, Sainz M, *et al*. Interactive indirect illumination using voxel cone tracing: a preview[C] //Proceedings of the Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2011: 1921-1930

[15] Panteleev A. Practical real-time voxel-based global illumination for current GPUs[OL]. [2024-11-09]. https://cgvr.informatik.uni-bremen.de/theses/finishedtheses/VoxelConeTracing/S4552-rt-voxel-based-global-illumination-gpus.pdf

[16] Aherne I, Davison R, Ushaw G, *et al*. Adoption of sparse 3D textures for voxel cone tracing in real time global illumination[C] //Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. Setúbal: SciTePress, 2020: 201-209

[17] Franke T A. Delta voxel cone tracing[C] //Proceedings of the IEEE International Symposium on Mixed and Augmented Reality. Los Alamitos: IEEE Computer Society Press, 2014: 39-44

[18] Sugihara M, Rauwendaal R, Salvi M. Layered reflective shadow maps for voxel-based indirect illumination[C] // Proceedings of the High-Performance Graphics. Goslar: Eurographics Association, 2014: 117-125

[19] Chen Y Y, Chien S Y. Lighting-driven voxels for memory-efficient computation of indirect illumination[J]. The Visual Computer, 2016, 32(6-8): 781-789

[20] Shanmugam P, Arikan O. Hardware accelerated ambient occlusion techniques on GPUs[C] //Proceedings of the Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2007: 73-80

[21] Mittring M. Finding next gen: CryEngine 2[C] //Proceedings of the ACM SIGGRAPH Courses. New York: ACM Press, 2007: 97-121

[22] Bavoil L, Sainz M, Dimitrov R. Image-space horizon-based ambient occlusion[C] //Proceedings of the ACM SIGGRAPH Talks. New York: ACM Press, 2008: Article No.22

[23] Bavoil L. Horizon-based ambient occlusion using compute shaders[OL]. [2024-11-09]. https://developer.download.nvidia.com/assets/gamedev/files/sdk/11/SSAO11.pdf

[24] Ritschel T, Grosch T, Seidel H P. Approximating dynamic global illumination in image space[C] //Proceedings of the Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2009: 75-82

[25] McGuire M, Mara M. Efficient GPU screen-space ray tracing[J]. Journal of Computer Graphics Techniques, 2014, 3(4): 73-85

[26] Sloan P P, Govindaraju N K, Nowrouzezahrai D, *et al*. Image-based proxy accumulation for real-time soft global illumi-

nation[C] //Proceedings of the 15th Pacific Conference on Computer Graphics and Applications. Los Alamitos: IEEE Computer Society Press, 2007: 97-105

[27] Nichols G, Shopf J, Wyman C. Hierarchical image-space radiosity for interactive global illumination[J]. Computer Graphics Forum, 2009, 28(4): 1141-1149

[28] Nichols G, Wyman C. Multiresolution splatting for indirect illumination[C] //Proceedings of the Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2009: 83-90

[29] Frisken S F, Perry R N, Rockwood A P, *et al*. Adaptively sampled distance fields: a general representation of shape for computer graphics[C] //Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 2000: 249-254

[30] Evans A. Fast approximations for global illumination on dynamic scenes[C] //Proceedings of the ACM SIGGRAPH Courses. New York: ACM Press, 2006: 153-171

[31] Epic. Mesh distance fields in unreal engine | unreal engine 5.4 documentation[OL]. [2024-11-09]. https://dev.epicgames.com/documentation/en-us/unreal-engine/mesh-distance-fields-in-unreal-engine

[32] Wright D, Narkowicz K, Kelly P. Lumen: real-time global illumination in unreal engine 5[OL]. [2024-11-09]. https://dev.epicgames.com/documentation/en-us/unreal-engine/global-illumination-in-unreal-engine

[33] Veach E, Guibas L J. Optimally combining sampling techniques for Monte Carlo rendering[C] //Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 1995: 419-428

[34] Kondapaneni I, Vévoda P, Grittmann P, *et al*. Optimal multiple importance sampling[J]. ACM Transactions on Graphics, 2019, 38(4): Article No.37

[35] Hey H, Purgathofer W. Importance sampling with hemispherical particle footprints[C] //Proceedings of the 18th Spring Conference on Computer Graphics. New York: ACM Press, 2002: 107-114

[36] Jensen H W. Importance driven path tracing using the photon map[C] //Proceedings of the Eurographics Workshop in Dublin Rendering Techniques'95. Heidelberg: Springer, 1995: 326-335

[37] Müller T, Gross M, Novák J. Practical path guiding for efficient light-transport simulation[J]. Computer Graphics Forum, 2017, 36(4): 91-100

[38] Vorba J, Karlík O, Šik M, *et al*. On-line learning of parametric mixture models for light transport simulation[J]. ACM Transactions on Graphics, 2014, 33(4): Article No.101

[39] Bitterli B, Wyman C, Pharr M, *et al*. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting[J]. ACM Transactions on Graphics, 2020, 39(4): Article No.148

[40] Talbot J F. Importance resampling for global illumination[D]. Provo: Brigham Young University, 2005

[41] Ouyang Y, Liu S, Kettunen M, *et al*. ReSTIR GI: path resampling for real-time path tracing[J]. Computer Graphics Forum, 2021, 40(8): 17-29

[42] Lin D Q, Kettunen M, Bitterli B, *et al*. Generalized resampled importance sampling: foundations of Restir[J]. ACM Transactions on Graphics, 2022, 41(4): Article No.75

[43] Robison A, Shirley P. Image space gathering[C] //Proceedings of the Conference on High Performance Graphics. New York: ACM Press, 2009: 91-98

[44] Dammertz H, Sewtz D, Hanika J, *et al*. Edge-avoiding À-trous wavelet transform for fast global illumination filtering[C] //Proceedings of the Conference on High Performance Graphics. Aire-la-Ville: Eurographics Association Press, 2010: 67-75

[45] Hanika J, Dammertz H, Lensch H. Edge-optimized À-trous wavelets for local contrast enhancement with robust denoising[J]. Computer Graphics Forum, 2011, 30(7): 1879-1886

[46] Gastal E S L, Oliveira M M. Adaptive manifolds for real-time high-dimensional filtering[J]. ACM Transactions on Graphics, 2012, 31(4): Article No.33

[47] Bauszat P, Eisemann M, Magnor M. Guided image filtering for interactive high-quality global illumination[J]. Computer Graphics Forum, 2011, 30(4): 1361-1368

[48] Zwicker M, Jarosz W, Lehtinen J, *et al*. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering[J]. Computer Graphics Forum, 2015, 34(2): 667-681

[49] Mehta S U, Wang B, Ramamoorthi R. Axis-aligned filtering for interactive sampled soft shadows[J]. ACM Transactions on Graphics, 2012, 31(6): Article No.163

[50] Mehta S U, Wang B, Ramamoorthi R, *et al*. Axis-aligned filtering for interactive physically-based diffuse indirect lighting[J]. ACM Transactions on Graphics, 2013, 32(4): Article No.96

[51] Yan L Q, Mehta S U, Ramamoorthi R, *et al*. Fast 4D sheared filtering for interactive rendering of distribution effects[J]. ACM Transactions on Graphics, 2015, 35(1): Article No.7

[52] Meyer M, Anderson J. Statistical acceleration for animated global illumination[J]. ACM Transactions on Graphics, 2006, 25(3): 1075-1080

[53] Delbracio M, Musé P, Buades A, *et al*. Boosting Monte Carlo rendering by ray histogram fusion[J]. ACM Transactions on Graphics, 2014, 33(1): Article No.8

[54] Nehab D, Sander P V, Lawrence J, *et al*. Accelerating real-time shading with reverse reprojection caching[C] //Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware. Aire-la-Ville: Eurographics Association Press, 2007: 25-35

[55] Walter B, Drettakis G, Parker S. Interactive rendering using the render cache[C] //Proceedings of the Eurographics Workshop in Granada Rendering Techniques'99. Heidelberg: Springer, 1999: 19-30

[56] Karis B. High-quality temporal supersampling[OL]. [2024-11-09]. https://advances.realtimerendering.com/s2014

[57] Chaitanya C R A, Kaplanyan A S, Schied C, *et al*. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder[J]. ACM Transactions on Graphics, 2017, 36(4): Article No.98

[58] Nvidia DLSS(deep learning super sampling)[OL]. [2024-11-09]. https://www.nvidia.com/en-us/geforce/technologies/dlss

[59] Liu E. DLSS 2.0-image reconstruction for real-time rendering with deep learning[OL]. [2024-11-09]. https://developer.nvidia.com/gtc/2020/video/s22698-vid

[60] AMD. FidelityFX™ super resolution[OL]. [2024-11-09]. https://www.amd.com/en/products/graphics/technologies/fidelityfx/su-

per-resolution.html

[61] Intel. Intel® Arc™-xe super sampling[OL]. [2024-11-09]. https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/arc/technology/xess.html

[62] Hadadan S, Chen S H, Zwicker M. Neural radiosity[J]. ACM Transactions on Graphics, 2021, 40(6): Article No.236

[63] Zheng C K, Huo Y C, Mo S H, *et al*. NeLT: object-oriented neural light transfer[J]. ACM Transactions on Graphics, 2023, 42(5): Article No.163

[64] Wu S Y, Kim S, Zeng Z, *et al*. ExtraSS: a framework for joint spatial super sampling and frame extrapolation[C] //Proceedings of the SIGGRAPH Asia Conference Papers. New York: ACM Press, 2023: Article No.92

[65] Li Z, Marshall C S, Vembar D S, *et al*. Future frame synthesis for fast Monte Carlo rendering[C] //Proceedings of the 48th Graphics Interface Conference. Montréal: Canadian Human-Computer Communications Society, 2022: 74-83

[66] Guo J, Fu X H, Lin L Q, *et al*. ExtraNet: real-time extrapolated rendering for low-latency temporal supersampling[J]. ACM Transactions on Graphics, 2021, 40(6): Article No.278

[67] Zhong Z H, Zhu J S, Dai Y X, *et al*. FuseSR: super resolution for real-time rendering through efficient multi-resolution fusion[C] //Proceedings of the SIGGRAPH Asia Conference Papers. New York: ACM Press, 2023: Article No.8

[68] Zhang H N, Guo J, Zhang J W, *et al*. Deep Fourier-based arbitrary-scale super-resolution for real-time rendering[C] //Proceedings of the ACM SIGGRAPH Conference Papers. New York: ACM Press, 2024: Article No.65

[69] Müller T, Rousselle F, Novák J, *et al*. Real-time neural radiance caching for path tracing[J]. ACM Transactions on Graphics, 2021, 40(4): Article No.36

[70] Greger G, Shirley P, Hubbard P M, *et al*. The irradiance volume[J]. IEEE Computer Graphics and Applications, 1998, 18(2): 32-43

[71] McGuire M, Mara M, Nowrouzezahrai D, *et al*. Real-time global illumination using precomputed light field probes[C] //Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2017: Article No.2

[72] Majercik Z, Guertin J P, Nowrouzezahrai D, *et al*. Dynamic diffuse global illumination with ray-traced irradiance fields[J]. Journal of Computer Graphics Techniques, 2019, 8(2): 1-30

[73] Lischinski D, Tampieri F, Greenberg D P. Combining hierarchical radiosity and discontinuity meshing[C] //Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques. New York: ACM Press, 1993: 199-208

[74] Majercik Z, Marrs A, Spjut J, *et al*. Scaling probe-based real-time dynamic global illumination for production[J]. Journal of Computer Graphics Techniques, 2021, 10(2): 1-29

[75] Hu J K, Yip M K, Alonso G E, *et al*. Efficient real-time dynamic diffuse global illumination using signed distance fields[J]. The Visual Computer, 2021, 37(9-11): 2539-2551

[76] Majercik Z, Müeller T, Keller A, *et al*. Dynamic diffuse global illumination resampling[C] //Proceedings of the ACM SIGGRAPH Talks. New York: ACM Press, 2021: Article No.24

[77] Halen H, Hayward K, Brinck A, *et al*. Global illumination based on surfels[OL]. [2024-11-09]. https://www.ea.com/seed/news/ siggraph21-global-illumination-surfels

[78] Boissé G, Meunier S, de Dinechin H, *et al*. Gi-1.0: a fast and scalable two-level radiance caching scheme for real-time global illumination[OL]. [2024-11-09]. https://arxiv.org/abs/2310.19855

[79] Dong H H, Wang G P, Li S. Neural parametric mixtures for path guiding[C] //Proceedings of the ACM SIGGRAPH. New York: ACM Press, 2023: Article No.29