arXiv:2303.04086v1 [cs.GR] 7 Mar 2023

NEPHELE：高真实感云端辐射渲染神经平台

HAIMIN LUO、张思远、Fuqiang Zhao、景浩天、王鹏浩、俞振晓、严冬雪、丁俊然、张博远、胡强、殷舒、Lan Xu 和 Jingyi Yu，上海科技大学，中国



图1. **NEPHELE 平台演示**。我们介绍 NEPHELE，一个用于高真实感云端辐射渲染的神经平台。我们将物体（例如**中国生肖**）表示为 i-NOLF，以实现超快速神经辐射渲染。借助定制的任务调度器，NEPHELE 能够利用多个 GPU 的算力，渲染由众多神经资产组成的复杂场景，而这无法通过单个 GPU 实现。NEPHELE 还会将照片级真实感的辐射度渲染结果分发给客户端，让所有人都能通过虚拟现实头显和手机等各种设备，轻松访问云端可共享的神经资产。

近年来，神经渲染（NR）领域取得了巨大进展，例如 NeRF，可用于照片级真实感的自由视点合成。然而，作为一种依赖单台计算机/GPU 的本地技术，即使是经过精心优化的 Instant-NGP 或 i-NGP，在高分辨率渲染时也无法达到实时性能，而且往往需要大量本地计算资源。本文转而采用云渲染，并介绍 NEPHELE，一个用于高真实感云端辐射渲染的神经平台。与现有神经辐射方法截然不同，NEPHELE 通过组合多个远程 GPU，提供更强大的渲染能力，并支持多人同时查看同一个 NeRF 场景，促进协作。NeRF 与云渲染的这种结合，自然需要一种轻量、实时且具备灵活可扩展性的神经渲染器。为此，类似地

作者地址：Haimin Luo；张思远；Fuqiang Zhao；景浩天；王鹏浩；俞振晓；严冬雪；丁俊然；张博远；胡强；ShuYin；Lan Xu；Jingyi Yu，上海科技大学，中国。

针对 i-NGP，我们提出 i-NOLF，利用不透明度光场，以每条光线仅查询一次的方式实现超快速神经辐射渲染。我们进一步通过几何代理重新构建 Lumigraph，以实现快速光线查询，随后使用小型多层感知机对局部不透明度光球进行建模，从而实现高质量渲染。我们还在 i-NOLF 中采用完美空间哈希，取代原始 i-NGP 中的暴力多重哈希，以增强缓存一致性。因此，与 i-NGP 相比，我们的 i-NOLF 在效率方面实现了一个数量级的性能提升，尤其适用于云渲染场景下的多用户、多视点设置。我们还针对 i-NOLF 表示定制了一个任务调度器，并采用光线级调度设计来维持渲染作业的韧性。我们也通过一个综合性云平台展示了方法设计的优势，该平台由一系列协同工作的模块组成，包括渲染农场、任务分配器、帧合成器以及详细的流传输策略。借助这一兼容神经渲染的云平台，我们

进一步通过一系列应用展示了云端辐射渲染的能力，应用范围从云端虚拟现实/增强现实渲染，到在多个用户之间共享 NeRF 资产，并允许 NeRF 资产自由组合成新的场景。

附加关键词与短语：神经渲染、云渲染

1 引言

随着神经渲染迅速发展并逐渐取代传统的基于物理的方法，人们对更易获取、更经济实惠的实用方案的需求也随之增长。基于辐射场的方法（例如 NeRF）通过采用定制的深度网络，以光线为基础同时对场景几何和外观进行建模，将渲染转化为通过网络进行光线查询。此类方法能够以极高的照片级真实感重现复杂的真实世界物体，甚至大型环境。然而，作为一种依赖单台计算机/GPU 的本地技术，即使是经过精心优化的 instance-NGP，在高分辨率渲染时也无法达到实时性能：对于单个 NeRF 对象，即使使用最新的 NV GeForce 4090，i-NGP 在 4K 分辨率下也只能勉强达到 10 fps。

本文采用云渲染，并提出 NEPHELE，即高真实感云端辐射渲染神经平台。简而言之，NEPHELE 使用远程服务器，而非单台本地计算机的处理能力，不仅能够通过整合多个 GPU 提供更强大的渲染能力，还能实现一系列前所未有的效果。例如，NEPHELE 支持渲染由众多 NeRF 对象组成的复杂场景，这类场景对单台计算机而言负担过重。更重要的是，它通过允许多人同时查看同一个 NeRF 场景来促进协作。此外，NEPHELE 使渲染过程能够在后台运行，从而释放用户的计算资源以执行其他任务，例如支持各种虚拟现实和增强现实体验。

传统云渲染主要基于光栅化渲染管线。它们本质上会将渲染过程拆分成更小的任务，再分发到多台服务器上。具体而言，首先将三维场景和资产划分为更小且易于管理的子集，每个子集都可在各个服务器上实时渲染。随后，渲染结果会发送回云平台，并在此处合成为最终图像。然而，将经典云端解决方案直接应用于神经辐射场渲染会面临多项挑战。首先，NeRF 中最小的“子集”是一个通常代表完整对象的神经网络，因此每个网络都需要分配一张显卡。即便如此，实时渲染仍要么牺牲分辨率 [Chen 等，2022a；Luo 等人 2022]，要么需要超高存储需求 [Chen 等，2022a；Sara Fridovich-Keil 和 Alex Yu 等人，2022；Yu 等人 2021]。因此，亟需一种轻量、实时的神经渲染器，以保持灵活性和可扩展性。其次，为了获得沉浸式体验，多个用户应能同时从不同视角查看同一个 NeRF 对象。为 i-NGP 中的单用户设计的哈希编码可能导致较高的未命中率，并降低渲染性能。最后，调度器应将多个渲染作业分配给最适合的显卡，以提高硬件利用率和渲染性能，并确保良好的数据一致性，从而能够同时查看同一个 NeRF 对象。但现有研究关注以作业为单位的调度策略，这会带来上下文切换开销，并导致显卡利用率低下。因此，更具针对性的调度器有助于保持渲染作业的弹性。

在 NEPHELE 中，我们首先开发了一种超高速神经辐射度渲染器，称为即时神经不透明度光场（i-NOLF）。i-NOLF 将不透明度光场[Vlasic 等人 2003] 与 i-NGP 相结合，采用每条光线仅查询一次的方案，大幅加快命中点估计，进而提升整体渲染速度。它类似于 Lumigraph [Buehler 等人 2001]：后者使用几何代理来快速进行光线查询，并采用额外的小型多层感知机对局部不透明度光球进行建模，以支持半透明性和镜面反射性等视图相关特征。通过使用高效的数据结构缓存视图无关属性，i-NOLF 的速度比 i-NPG 提升了半个数量级以上，并且能够以每秒 30 帧以上的速度进行 4K 分辨率渲染。为进一步突破多分辨率哈希编码 [Müller 等人 2022],的局限，我们发现，在多用户、多视点设置下，大量缓存流量是由访问均匀分布在哈希表各层的短特征造成的。为此，我们采用完美空间哈希 [Lefebvre 和 Hoppe 2006] 重新组织内存分布，以提升缓存一致性，并开发了单分辨率编码方案来减少缓存流量。结合这两种方案后，i-NOLF 在多用户设置下的速度比 i-NGP 快一个数量级。

在系统层面，高效的多用户云渲染器依靠智能的任务分发/调度和结果汇聚，在优化响应时间的同时提高渲染农场中 GPU 的利用率。我们提出了一种光线级渲染机制，将多用户渲染任务视为一组不加区分、带有深度的光线。通过这种方式，我们的调度器能够依据光线级指标调度多个渲染任务，从而保持较高的 GPU 利用效率。其次，我们提出了一种包含集中式调度器的光线级调度设计。主节点合并具有共性的任务，以利用缓存，并按正确顺序组合结果。

NEPHELE 旨在通过云端分发辐射度渲染的强大能力，正如阳光穿过云层照耀大地。神经渲染的目标应用之一，是让用户在增强现实或虚拟现实环境中获得沉浸式体验。为了将 NEPHELE 与 Apple ARKit [Oufqir 等人 2020]以及 WebXR 设备 API [Maclntyre 和 Smith 2018],等成熟解决方案无缝连接，我们采用基于流式传输的方案：在云服务器上执行计算密集型任务，并将结果作为视频流发送到客户端。为了保持低延迟和交互性，我们采用 Web 实时通信（WebRTC） [Suciu 等人 2020]，使服务器和客户端能够彼此直接通信。在我们的云端 AR 应用中，移动设备使用 ARKit 持续向云服务器发送用户交互数据，例如变换矩阵、内参矩阵和检测到的平面信息。云服务器根据交互数据，在纯绿色背景上渲染内容，并将渲染帧压缩成视频流，通过 WebRTC 传输到移动设备。移动设备解码视频帧，并通过色度键控进行背景抠像，使解码帧中特定范围的区域变为透明

。最后，我们将这些帧叠加到摄像头画面上，带来融合物理世界与数字世界的沉浸式体验。在我们的云端 VR 应用中，我们使用设备分辨率和摄像机投影矩阵初始化云端任务，然后持续接收设备当前的视图和位姿。云端任务为每个视图渲染一个场景，并通过 WebRTC 返回合成帧和深度信息。Web 应用根据返回的数据创建纹理、材质和网格，并使用网格渲染场景。最后，我们展示 NEPHELE 如何通过云端分发辐射度渲染的强大能力，以支持多用户场景：从让多个用户体验和共享 NeRF 资产，到允许 NeRF 资产自由组合成新场景。

总而言之，我们的主要贡献如下：

- 我们提出了 NEPHELE，这是一种云渲染平台，旨在通过云端分发辐射度渲染能力，从而提供更强大的渲染能力。
- 我们提出了一种超快速神经辐射渲染器 i-NOLF，将不透明度光场和完美空间哈希应用于 i-NGP。它的性能显著优于 i-NGP，尤其适用于多用户多视点设置。
- 我们为 i-NOLF 表示量身设计了一种任务调度器，并采用光线级调度设计，以维持渲染作业的韧性。
- 我们通过一系列应用展示了云端辐射渲染的能力，例如云端 VR/AR 渲染，或共享和组装各种 NeRF 资产。

2 相关工作

神经辐射渲染。 近年来，神经辐射场（NeRF） [Mildenhall 等人 2021] 提出的技术及其扩展 [Barron 等人 2021年；Chen 等人 2021年；Luo 等人 2021年；Verbin 等人 2022；Wang 等人 2021] 为照片级真实感新视角合成带来了巨大潜力。其动态扩展[Liu 等人 2021年；Peng 等人 2021年；Pumarola 等人 2020年；Tretschk 等人 2021年；Zhang 等人 2021a；Zhao 等人 2022] 利用 NeRF 体渲染技术，专注于动态场景。一些工作 [Boss 等人 2021年；Pan 等人 2021年；Srinivasan 等人 2021年；Zhang 等人 2021b] 通过光照分解扩展了 NeRF，使 NeRF 具备重新照明效果。其他后续工作 [Chen 等，2022b；Lombardi 等人 2021年；Müller 等人 2022；Sara Fridovich-Keil 和 Alex Yu 等人 2022；Sun 等，2022；Wang 等人 2022b；Yu 等人 2021] 提升了 NeRF 的训练和渲染速度，使 NeRF 表示更加高效且实用。例如，PlenOctrees [Yu 等人2021] 将 NeRF 的连续表示转换为离散的八叉树表示。借助这种高效的 GPU 八叉树实现，它无需进行任何多层感知机评估即可实现 > 3000× 渲染加速。i-NGP [Müller 等人 2022] 进一步结合多分辨率哈希编码和微型多层感知机，提升了训练速度。它帮助 i-NGP [Müller 等人 2022] 将训练时间缩短至 5 秒，同时保持 > 3000× 相较于原始 NeRF 的渲染加速效果。然而，这些方法仍然需要高端 GPU 设备来支持计算密集型的训练和渲染过程。MobileNeRF [Chen 等，2022a] 提出了一种新的 NeRF 表示，该表示基于一组带纹理的多边形构建，这些纹理由二值不透明度和特征向量组成。它可以在手机和轻薄笔记本电脑等移动设备上高效渲染新视角图像。然而，它仍然依赖高端 GPU 来优化这种基于带纹理多边形的 NeRF 表示。此外，这种提出的基于带纹理多边形的表示也失去了视角相关效果。现有大多数相关方法都专注于单个物体或场景。较少有方法 [Guo 等人 2020年；Niemeyer 和 Geiger 2021年；Zhang 等人 2021a] 使 NeRF 表示具备组合性。为此，我们旨在构建一个 NeRF云渲染平台（NEPHELE），消除用户对高端设备的依赖，让 NeRF 技术更加普及。所提出的 NEPHELE 在虚拟现实/增强现实应用和全息显示器（如 Looking Glass）中保持 NeRF 的实时渲染。此外，NEPHELE 还将组合性赋予该云平台上的每个 NeRF 对象。

***GPU*集群的任务调度。** 许多研究都致力于 GPU 集群调度器。SLAQ [Zhang 等人 2017] 和 Optimus [Peng 等人2018]通过估计各作业的未来步骤来调度分布式作业，以尽可能降低平均计算时间。Gandiva [Xiao 等人 2018] 和 THEMIS [Mahajan 等人 2020] 则更进一步，利用 GPU资源上的时间切片机制考虑完成时间公平性。其他一些调度器致力于根据作业优先级制定放置策略，以改善本地性性能。例如，Philly [Jeon 等人 2019] 基于生产集群的跟踪数据分析，尝试微调调度策略，以缩短平均作业完成时间。Tiresias [Gu 等人 2019] 利用作业已获得的服务时间构建优先队列进行调度。ONES [Bian 等人 2021] 确定每个作业的批大小，并为弹性工作负载提出了一种在线进化调度器。一些早期研究提出基于贪心策略的调度器，以最大限度地提高系统整体利用率，例如 OASiS [Bao 等人 2018]。由于应用多种多样，所有这些工作都以作业为单位关注调度策略。此外，将整个作业放置在专用 GPU资源上，可通过隔离保持良好的数据一致性和保护性，但这会给 GPU 集群带来上下文切换开销，并降低 GPU资源的利用率。我们注意到，渲染作业与一般基于GPU的应用之间存在差异：多个渲染作业可以共享一定量的数据，尤其是在从不同角度观察相同对象时。基于这一观察，我们提出的 NEPHELE 系统提供了一种放置策略，将渲染作业按光线分组处理。借助更细粒度的渲染方式，NEPHELE 在调度作业时会综合考虑多个因素（即公平性、响应性、作业完成时间和总利用率）。

流媒体。 如今，云端渲染技术广泛应用于渲染农场，有望降低客户端的硬件需求，并使用户摆脱不断进行硬件升级的负担。然而，如果没有针对云端渲染应用设计的适当流媒体解决方案，这种可能带来革命性变革的计算范式可能会彻底失败。交互延迟是云端渲染面临的主要问题之一。用户交互会被发送到云端，在那里根据用户交互渲染场景

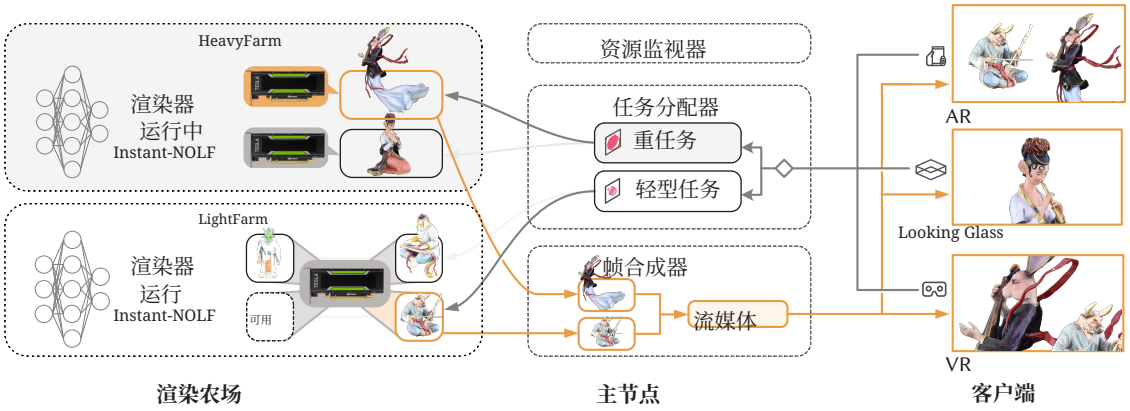

图2。**我们提出的NEPHELE系统概览**。该流程图展示了在云平台中渲染场景的过程：用户使用客户端连接至**主节点**，主节点创建会话并建立WebRTC连接。用户通过提交其场景和视图对象来发出请求。随后，**任务分配器**根据命中密度网格的光线大致数量，将任务分类为轻型任务或重型任务。轻型任务被路由至轻量渲染农场进行渲染，而重型任务则被路由至重型渲染农场。**渲染农场**将任务渲染为帧，随后**帧合成器**将其组装，以匹配用户的请求。最后，**帧合成器**将完成的帧发送至客户端供查看。实线黑色矩形表示有效的*GPU*资源。橙色矩形表示一个*GPU*被重任务（*HeavyFarm*）独占，或*GPU*的一部分被轻型任务占用。虚线矩形表示可用于新分配任务的*GPU*资源。

。云端的渲染帧应尽快编码并传送到客户端。Apple HTTP实时流媒体（HLS） [Pantos 2022] 和基于 HTTP 的 MPEG 动态自适应流媒体（DASH） [Stockhammer 2011] 是最常用的两种 HTTP自适应流媒体协议。它们通过将视频拆分成较小的文件（称为分段）来工作，客户端可以立即下载这些文件。这样会导致大约数秒的端到端延迟。因此，HLS 和 DASH 不适用于云端渲染。

为支持这些低延迟且具交互性的使用场景，通常会采用实时通信（RTC）解决方案。Web 实时通信（WebRTC）是由谷歌发起的开源项目，采用简单的点对点架构来实现高效通信 [Suciu et al. 2020]。WebRTC 具有低延迟特点，同时还包含一套完整的协议设计和服务质量保障机制。随着 WebRTC 技术的发展，许多相关应用也因其快速且稳定的传输速度而应运而生，例如远程XR [Lee等人 2021; Vikberg et al. 2021], 远程医疗 [Ha et al. 2020], 以及云游戏 [Di Domenico et al. 2021] 。我们的工作同样利用了 WebRTC 的特性，使用户能够体验低延迟、高质量的云渲染服务。

3 概述：NEPHELE架构

3.1 设计目标

首先，NEPHELE 应支持多个用户同时进行渲染任务，让多个用户能够随机访问 GPU 集群以查看不同角度的画面。其次，NEPHELE 应将独立的渲染任务视为一组具有深度信息、彼此无差别的光线，从而忽略任务边界，以便更好地重复利用数据。第三，NEPHELE 应以光线组为单位将任务调度到 GPU 上，从而提高利用率并缩短响应时间。

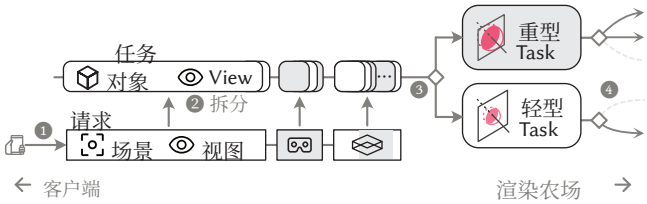

图 3. **任务分配器的功能**。它接收来自客户端的请求，其中包含场景信息和所需视图。随后，请求会被拆分为单个任务，每个任务包含一个带角度的对象。系统根据预计与密度网格相交的光线数量，将任务分为轻型任务或重型任务。轻型任务会发送到专用的轻量渲染农场进行渲染，而重型任务则会被路由至重型渲染农场处理。

3.2 NEPHELE架构

图 2 展示了 NEPHELE 的架构。NEPHELE 由三个主要组件构成：RenderFarm 以光线组为单位执行渲染任务；主节点监控渲染任务的状态，将任务分配给 RenderFarm 中的目标 GPU，并按正确顺序组合渲染结果；客户端代表各类设备，用于提交渲染作业并显示结果（例如 PC、手机、增强现实设备、虚拟现实设备等）。

渲染农场 强调，用户可以**共享** 同一**进程** 来渲染相同的 NeRF 模型，从而实现最高的硬件利用率；通过在多个用户之间复用相似的渲染结果，减少不必要的重复计算，降低每位用户的计算成本。相比之下，在传统的云游戏渲染或云渲染农场中，每位用户的渲染任务都在不同的进程中运行，甚至使用完全隔离的硬件。

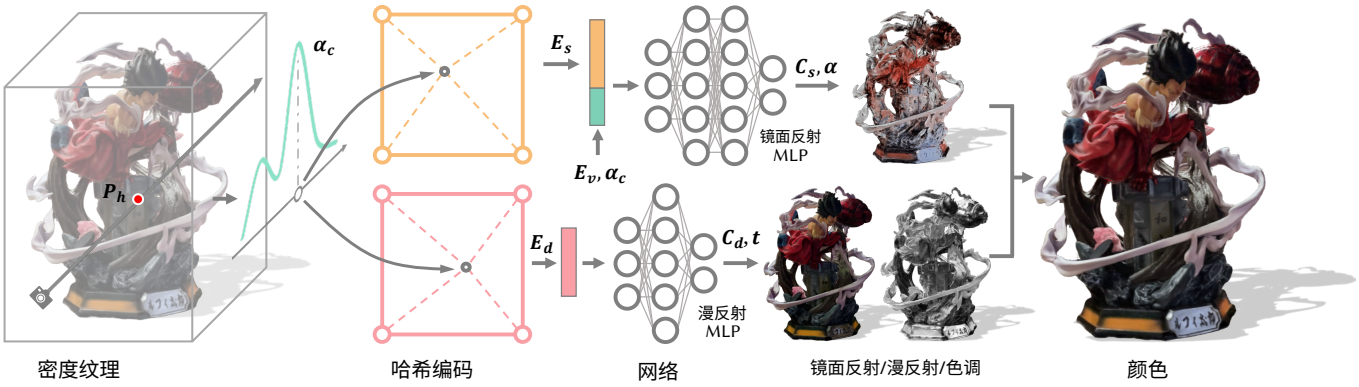

图4. **我们使用 i-NOLF 渲染中国神话人物哪吒的示意图**。对于给定的光线，我们首先在缓存密度纹理上使用光线步进找到命中点坐标 P_h，并累积该光线的粗粒度不透明度 α_c。随后，使用镜面反射哈希编码器将得到的坐标编码为特征向量 E_s。我们将 E_s 与视角方向编码 E_v 及 α_c 拼接起来。然后，镜面反射多层感知机将得到的特征映射为镜面反射颜色 C_s 和不透明度 α。此外，我们使用另一个漫反射哈希编码器将 P_h 编码为 E_d，然后采用漫反射多层感知机预测漫反射颜色 C_d 和镜面反射色调 t。最后，将漫反射颜色和镜面反射颜色组合起来，得到最终的出射辐射。

主节点 负责监控渲染农场的状态，将渲染任务分配给渲染农场中的目标 GPU，并按照正确的顺序为客户端 组合基于光线的渲染结果。通常，资源监视器会监测渲染农场中 GPU 的利用率，并通知任务分配器将渲染任务分发给目标 GPU。

主节点中的任务分配器使单个图形处理器能够运行大量低精度 NeRF 实例，并使多个图形处理器能够运行少量高精度实例。任务分配器首先将渲染作业转换为一组带有角度的光线，并根据视觉覆盖范围将光线归类为重型和轻型任务。重型任务表示大量光线命中物体，而轻型任务则意味着只有少量光线命中物体。重型任务可能是用户近距离注视一个小物体，或观察一个大物体；轻型任务则可能是用户俯瞰一个大物体，或观察一个小物体。调度细节见第6节。

客户端 是由不同用户提交的多个渲染作业。这些作业可以是单个访客使用增强现实（即单用户多对象）或头戴式显示设备在博物馆中游览文物，也可以是一群游客在画廊中观赏同一件艺术品（即多用户单对象）。由于 NeRF模型初始化和上下文切换的开销，现有渲染技术通常会为单个作业独占一个或多个 GPU。对于单个用户或多个用户而言，渲染场景中的大量对象颇具挑战，因为这需要难以负担的 GPU资源。虽然我们可以将每个模型分别放置在 GPU 上进行渲染，再将结果合成到大型场景中，但 GPU之间上下文切换和通信的开销过高，无法提供经济且及时的 NeRF 渲染体验。

此外，当单个用户试图同时查看不同对象，或多个查看器同时查看同一对象时，由此产生的多个渲染任务会同时随机访问 GPU。这会导致响应时间无法接受，因为云调度器会将所有渲染任务排入 GPU 队列。新提交的渲染任务必须等待，直到有足够的 GPU资源释放出来。我们注意到，在不同的查看任务中，无论是单用户多对象还是多用户单对象的情况，都有相当数量的光线可以共享。如果多个渲染作业能够共享表示相似视角的数据，我们就能减少冗余的 GPU 渲染工作，并充分利用 GPU 集群处理更多作业。这要求渲染农场采用更细粒度的渲染算法执行渲染作业，使主节点能够以光线组为单位而非以任务为单位调度任务。

4 加速 I-NGP

作为 NEPHELE 渲染农场的核心，快速的神经渲染器对于尽可能缩短渲染逼真图像所需的时间、满足用户需求至关重要。我们着手加速近期的实时神经渲染器 i-NGP。首先，我们通过开发一种新颖的完美空间哈希编码，提高了 i-NGP 的内存请求效率。随后，我们将不透明度光场与 i-NGP 结合，提出了一种超快速渲染器，通过执行每条光线一次查询的渲染来避免冗余的计算开销。

4.1 完美空间哈希编码

多分辨率哈希编码的局限性。回顾一下，在 i-NGP [Müller 等人 2022], 的方法中，多分辨率哈希编码在粗分辨率下采用均匀网格编码，而在细分辨率下则采用哈希编码。将不同分辨率下的短特征向量拼接成最终的特征向量，从而得到一个内存占用可控的多分辨率特征向量。

然而，利用图形处理器实现这种编码效率低下。根据哈希函数的性质，对哈希表的内存访问会均匀分布在整个哈希表中。因此，如果哈希表无法装入缓存，就会发生缓存抖动，导致大量内存与缓存之间的数据流量。此外，每个层级都会独立请求内存，但只使用返回的缓存行中的一小部分，浪费内存带宽。

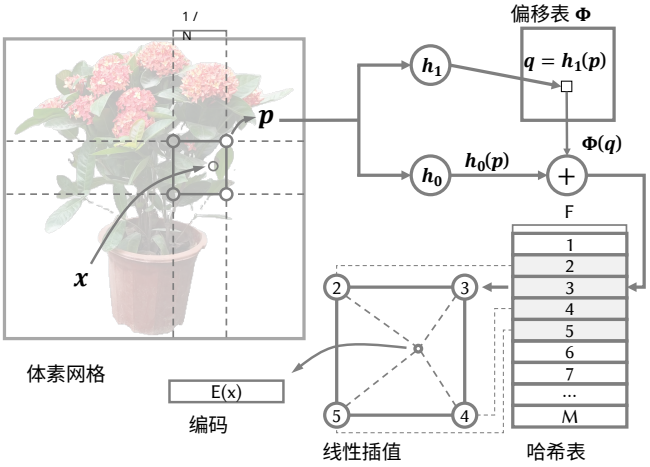

图 5. **二维完美空间哈希编码示意图**。对于给定的输入坐标 x，我们找到 x 所在体素的各个角点，并使用两个哈希函数 h_0 和 h_1 将这些角点的整数坐标 p 分别哈希为哈希表的基索引以及偏移表的索引 q Φ。我们根据得到的角点索引 h_0（p）+ Φ（q），从哈希表中查找对应的 F 维特征向量，并通过三线性插值生成最终编码 E（x）。请注意，空间上相邻的点会被映射到哈希表中相邻的槽位，且不会发生哈希冲突，从而最大限度地提高缓存一致性。

完美空间哈希。因此，为了实现缓存一致性并更好地利用内存带宽，我们提出了一种新颖的单分辨率完美空间哈希编码，如图 5 所示。完美空间哈希（PSH）是指针对静态数据（例如占据场）预先计算哈希函数，以实现零哈希冲突。我们采用以下形式的哈希函数 [Lefebvre 和 Hoppe 2006]

$$h(\mathrm{p}) = h_0(\mathrm{p}) + \Phi(h_1(\mathrm{p})) \tag{1}$$

其中，Φ 是一个偏移查找表，通过受 [Fox 等人 1992] 启发的启发式方法构建，旨在尽量减少哈希冲突。借助偏移表 Φ，PSH 倾向于找到一种映射方式，使空间上相邻的点被映射到哈希表中相邻的槽位，从而创造利用缓存一致性的机会。

与哈希编码类似，对于任意位置 x，我们将其乘以网格分辨率 N 并取整，以找到顶点坐标 $\mathrm{p} = \lfloor N * \mathrm{x} \rfloor$。然后，使用 PSH 提取网格各顶点对应的特征向量。最后，通过三线性插值得到结果特征向量。为了减少内存请求的数量，我们仅使用单一分辨率的特征向量。这样，每个采样点只需发出一次内存请求，并且只会用到返回的缓存行的一半。

i-NGP 的实现。我们用提出的 PSH 编码替换多分辨率哈希编码，以评估性能。具体而言，我们执行轮廓重建（SfS）算法，获取粗略的占据场，并据此计算偏移表 Φ。我们在不同 PSH 编码分辨率下，以及使用原始 i-NGP，在场景“ficus”上训练模型。如表 1 所示，分辨率越高，渲染质量越好；总体而言，其性能与哈希网格相当。然而，在内存请求效率（即 L2 缓存命中率）方面，在测试高并发随机渲染请求时，PSH 编码的表现显著优于哈希网格。在 4K 分辨率下，渲染速度提升了 20%，满足云渲染场景的需求。

表 1. **不同分辨率下 PSH 编码的性能**。我们通过将 Instan-NGP 原有的哈希网格编码替换为 PSH 编码，并在场景“ficus”上训练模型，对 PSH 编码进行评估。除渲染质量外，我们还通过 L2 缓存命中率和渲染随机 4K 分辨率图像的速度来评估内存效率。此外，我们还报告了编码器的参数数量。

方法	峰值信噪比↑	结构相似性↑	L2↑	大小（MB）↓	时间（毫秒）↓	FPS↑
哈希网格	32.90	0.892	72.82	12.20	69.54	14.38
PSH-128	31.61	0.960	94.18	1.37	56.09	17.83
PSH-256	32.31	0.967	94.33	7.63	56.30	17.76
PSH-512	32.40	0.968	94.23	51.25	58.55	17.08

4.2 即时神经不透明度光场

NeRF（包括 i-NGP [Müller 等人 2022]）的主要缺点是，即使采用提前停止加速策略，渲染一个像素也需要数百次神经网络前向传递。这会导致 GPU 内存 I/O 操作冗余，并在光线步进过程中产生计算开销，成为云端场景中进一步加快渲染过程的主要瓶颈。一种直接的方法 [是 Wang 等人 2022a] 提出的，将场景表示为神经光场（NeLF），以实现每个像素仅查询一次的渲染。这类方法通常需要一个庞大的全连接神经网络，在不依赖几何的情况下编码整个光场；这与 i-NGP 的核心设计理念背道而驰，即在高效空间哈希编码的前提下使用较小的神经网络。此外，哈希网格通常能够有效编码局部空间属性，却无法逼近全局光场函数。为此，我们尝试将场景表示为局部神经不透明度光场，以同时利用空间哈希编码和光场的优势。

不透明度光场。早期的基于图像的渲染技术本质上旨在用表面光场来建模全光函数 [Chen 等人 2018；Wood 等人 2000]，该光场将位于已知网格表面上的点的出射辐射编码为光球面。为摆脱物体几何复杂度带来的限制，不透明度光场 [Vlasic 等人 2003] 将表面光场与视角相关形状相结合，并采用不透明度包壳 [Matusik 等人 2002]。不透明度包壳使用代理几何体表面点的视角相关不透明度来建模精细的几何细节，例如羽毛。粗略地说，不透明度光场可以表示为一个函数：

$$L : M \times S^2 \rightarrow RGBA \tag{2}$$

其中，M 为表面网格，S^2 表示单位向量球面。与使用显式网格和视角相关纹理不同，我们将连续不透明度光场表示为一个五维函数，该函数以三维位置 $\mathrm{p} = (x, y, z)$ 和二维视线方向（θ，ϕ）为输入，并输出出射辐射 $\mathrm{c} = (r, g, b)$ 以及不透明度 α

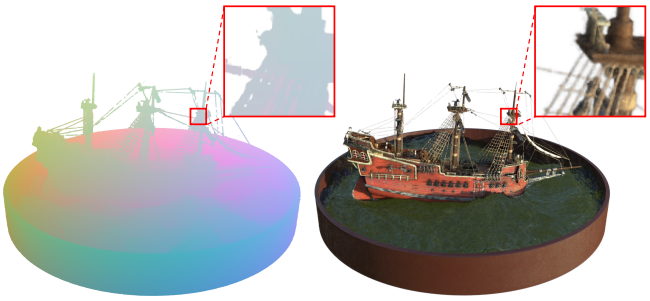

图 6. **估计得到的命中点图（左）和渲染的颜色图像（右）**。尽管失去了精确的几何细节，我们的命中点估计策略仍能在很大程度上消除代理形状造成的歧义。我们的 i-NOLF 还能进一步恢复高质量的几何和外观细节。

沿一条光线的值。与 i-NGP 类似，我们采用哈希编码器 E_s 将输入位置编码为特征向量 $\mathrm{E}_s(\mathrm{p})$，并使用一个浅层神经网络 F_s 将三维笛卡尔空间中归一化的视线方向 v 映射为视角相关颜色和不透明度。

$$F_s : (\mathrm{E}_s(\mathrm{p}), \mathrm{v}) \rightarrow (\mathrm{c}_s, \alpha) \tag{3}$$

请注意，渲染这样一个连续的不透明度场非常高效：对于一条光线，我们只需查询一次神经网络。

命中点估计与不透明度细化。为了更好地利用瞬时神经网络近似不透明度光场，必须尽可能使空间上相邻采样点的光球面保持相似。设想最糟糕的情况：以物体的边界框作为代理几何体来学习不透明度光场。两个相邻点经过空间编码后得到的特征向量相似，但其光球面却大不相同。使用浅层多层感知机很难近似这种近乎一对多的映射。

为此，我们试图利用 i-NGP 密度场中的几何先验来估计光线-物体命中点。回想一下，原始 NeRF [Mildenhall 等人 2021]通过对样本深度进行加权混合来估计光线的命中深度：

$$\begin{aligned} w_i &= T_i(1 - exp(-\sigma_i \delta_i)), \\ T_i &= exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j) \end{aligned} \tag{4}$$

其中，σ_i 是密度，δ_i 是相邻样本之间的距离。这种估计方法适用于不透明表面，但会产生嘈杂的命中点，尤其是在物体边界和毛发物体上。为了解决这一问题，我们采用一种经验策略，在光线步进过程中将混合权重最大的样本点 w_i 视为命中点。图 6 展示了估计的命中点图，该图有助于优化神经不透明度辐射场。

同时，为了便于近似不透明度，我们将样本权重累积到光线的粗粒度不透明度 α_c中，并将其输入 F_s 进行细化，得到最终不透明度 α，这与 ConvNeRF [Luo 等人 2021]的方法类似：

$$F_s : (\mathrm{E}_s(\mathrm{p}), \mathrm{v}, \alpha_c) \rightarrow (\mathrm{c}_s, \alpha) \tag{5}$$

漫反射与镜面反射分量。我们进一步将出射辐射度函数解耦为与视角无关的漫反射分量和与视角相关的镜面反射分量，这与中值去除 [Wood 等人 2000] 的策略类似，可有效地在表面光场压缩过程中保留镜面反射效果。关键观察是，镜面反射分量是从丰富的漫反射分量中减去后得到的剩余辐射度，因此应当更易压缩 [Wood 等人 2000]，也更易由神经网络学习 [Chen 等人 2018]。

与 RefNeRF [Verbin 等人 2022], 类似，我们采用额外的空间哈希编码器 E_d 和神经网络 F_d 来编码漫反射辐射函数。该函数仅以位置为输入，并预测漫反射颜色 c_d 和镜面反射色调 s，

$$F_d : \mathrm{E}_d(\mathrm{p}) \rightarrow (\mathrm{c}_d, t) \tag{6}$$

出射辐射可以通过以下形式将它们组合起来获得：

$$\mathrm{c} = \mathrm{c}_d + t \cdot \mathrm{c}_s \tag{7}$$

训练。为了联合学习哈希网格 E_s,E_d和神经网络 F_s, F_d的参数，我们利用 L^2 损失约束辐射度和不透明度，使其分别接近真实颜色 c^{gt} 和 alpha 值 α_{gt}。

$$\mathcal{L} = \frac{1}{N} \sum_{i}^{N} (\|\mathrm{c} - \mathrm{c}_{gt}\|^2 + \|\alpha - \alpha_{gt}\|^2) \tag{8}$$

其中，N表示一个批次中根据可优化误差图采样的光线数量。我们使用 Adam 优化器来优化神经不透明度光场，参数设置与 i-NGP 相同。

密度纹理立方体。请注意，命中点估计过程对于有效训练和渲染都是必需的，这意味着在光线步进过程中，需要进行多次耗时的网络查询来计算采样点的密度。与 i-NGP 相比，这会导致训练速度和渲染速度变慢，并带来额外的神经不透明度光场查询开销。为了解决这个问题，一种直接的方法是将连续密度场缓存为稠密密度网格，以便快速读取，但高内存占用限制了网格分辨率。相比之下，我们将密度场表示为一组小型稠密密度纹理立方体，从而跳过空空间，实现紧凑存储。具体而言，我们首先构建一个低分辨率的基础均匀索引网格，例如 $b = 128$，将落入有效网格单元的查询映射到有效立方体表中的对应地址。表中的每个立方体实际上又是一个分辨率为 $r = 16$ 的均匀网格，用于存储查询到的密度。借助缓存的密度立方体，命中点估计过程（主要是光线步进）会受到内存访问限制，因此我们进一步将立方体移至纹理内存，以提高缓存一致性，并利用硬件加速的线性插值来提升性能。至此，借助密度纹理立方体，我们可以在不到 $0.1ms$ 的时间内，准确获取用于渲染 1920×1080 分辨率图像的光线-物体命中点。

漫反射纹理立方体。由于采用解耦网络设计，利用两个独立的网络分别对漫反射和

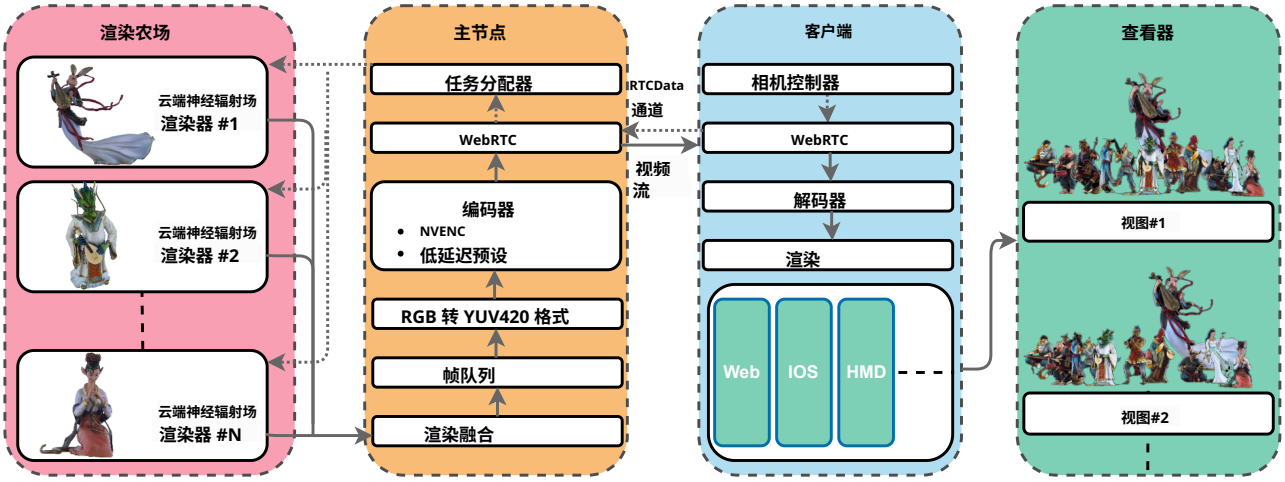

图 7. **云平台与数据流传输示意图**。为满足 NEPHELE 对低延迟和交互性的要求，我们采用 WebRTC 作为流传输协议，以高效交换信息。渲染场景时，客户端摄像机控制器每隔 20 毫秒通过 RTCDataChannel 向 NEPHELE 的主节点发送用户交互数据。随后，主节点中的任务分配器将渲染任务分配给渲染农场的计算子节点。所有子节点的渲染结果都会同步至主节点的渲染融合模块，并依据深度信息进行场景间融合。渲染帧使用启用低延迟预设的 NVENC 编码器压缩为比特流。最后，压缩后的比特流通过 WebRTC 传输至客户端进行解码和渲染。

镜面分量进行编码，训练完成后，我们可以进一步将漫反射网络的输出缓存为类似于密度纹理立方体的漫反射纹理立方体。需要注意的是，与密度纹理不同，我们只缓存包围所有可能命中点的体素的漫反射纹理，以减少内存使用量。这样，我们可以通过跳过两次网络查询来避免额外的时间开销，从而进一步提升渲染速度。

采用 *PSH* 编码的实现。鉴于表 1 所示，PSH 编码能够以略微降低渲染质量为代价，显著提升内存效率和渲染速度，下面介绍我们在实现中平衡其优缺点的方案。与缓存漫反射纹理的策略类似，我们构建初始哈希表，仅存储命中点周围体素的特征。这样一来，我们便能采用更精细的网格分辨率，以提升建模能力，同时将参数规模控制在可接受的范围内。此外，由于漫反射分量可以缓存，在线查询的只有镜面网络，因此我们仅对镜面编码器 E_s 采用 PSH 编码，而漫反射编码器仍使用哈希网格。这也符合将漫反射与镜面反射分量解耦的初衷：丰富的漫反射纹理需要更强的建模能力。

5 云平台

5.1 主渲染器与合成

传统的神经辐射场运行在用户主机设备上，这要求用户提供高端 GPU 设备来进行 NeRF 渲染。而且，该方案每次仅支持渲染一个场景，极大地限制了 NeRF 渲染体验。我们的目标是让每位用户都能体验 NeRF 渲染，而无需配备高端 GPU 设备。为此，我们提出了一种云渲染系统

该系统由负责部署 NeRF 任务调度的主节点和提供高性能计算能力的子节点组成，共同构成 NeRF 云平台。此外，云平台上的所有 NeRF 主体都可以自由组合，构建 NeRF 场景。计算子节点负责渲染各个 NeRF 主体的颜色和深度信息，所有子节点的渲染结果都会同步至主节点。随后，主节点根据深度信息进行场景间融合。最后，通过我们的流传输策略，将渲染结果流传输至用户的多屏终端。

5.2 流传输策略

对于 WebRTC 点对点通信，我们实现了一个信令服务器，用于交换媒体和网络元数据，以在云端与客户端之间建立对等连接。当客户端向信令服务器发送请求以体验云渲染服务时，信令服务器会为客户端分配 NEPHELE 中具有服务资源的主节点。通过发现与协商过程建立连接。随后，客户端摄像机控制器模块会频繁采样用户的位置和朝向，并将位置与视线方向转换为仿射变换矩阵。我们使用 WebRTC 的 RTCDataChannel [Jesup et al. 2015] 每 20 毫秒将该矩阵直接从客户端传输到主节点。由于使用 RTCDataChannel 传输数据时没有中间服务器且跳数更少，因此可以降低传输延迟。借助该矩阵，主节点会将渲染任务分配给渲染农场中的目标 GPU，并按照正确的顺序合成客户端所需的基于光线的渲染帧。最终，渲染帧被发送至编码器，以生成编码数据包。

NVIDIA GPU 配备了一种基于硬件的编码器（称为 NVENC），可提供完全加速的视频编码 [Patait and Young 2016],，因此我们使用由 NVENC 实现的 H.264 编解码器作为主节点配备的编码器。由于完整的编码过程均卸载至 NVENC，图形引擎和中央处理器可以用于其他应用，例如渲染。对于典型的端到端流媒体场景，为了实现低延迟，我们还通过选择采用 IPPP结构的图像组、不进行前瞻，以及针对给定比特率和可用信道带宽采用尽可能低的 VBV缓冲区，来最大限度地降低编码和解码延迟。在 IPPP 编码结构中（不包含 B帧），所有 P帧均采用前向预测，不使用后向预测或双向预测，这将降低编码和解码延迟，因此适用于我们的场景。由于渲染帧采用 RGB格式，我们将其转换为 YUV420格式，以实现高效压缩。压缩后的比特流会被封装为 RTP数据包、加密，并通过 WebRTC 以低延迟传输至客户端，使客户端能够在交互后实时接收所需帧。接收到的帧可以立即解码并显示，无需缓冲区。

6 多用户调度

我们讨论主节点如何应用调度策略，将多个渲染作业分配到渲染农场中的一组重型和轻型任务中。该策略适用于两种主要的渲染场景：单个用户在预定义场景中查看多个对象，以及多个用户从相邻角度查看同一对象。

6.1 单用户多对象调度

为单个用户提供查看大型场景中众多对象的服务，最简单的方案是分别渲染每个对象，并将它们合成为最终视图。在这种情况下，主节点会将用户的摄像机视图传递给渲染农场中的所有 NGP 模型（下文称为渲染器），让每个渲染器独立生成帧图像和深度信息，收集这些数据并将其合成到一个大型场景中，然后将结果发送回用户。然而，该方案面临两个障碍：1）主节点的输入/输出带宽限制，无法处理同时进行的图像传输。这会导致单个用户在大型场景中查看多个对象时，渲染响应时间无法接受。2）

由于大型场景中每个 NGP-NOLF 对象所命中的光线数量（记为 *nhit*）各不相同，因此 RendFarm 中不同 GPU 上的工作负载并不均衡，对渲染资源的需求也各不相同。若简单地为不同对象分配相同的渲染资源，会导致较小图像浪费 GPU 算力，而较大图像则产生较长的尾延迟。

我们提出了一种负载均衡调度策略，将对象分为两组：较小对象和较大对象。主节点将较大对象分配给 HeavyFarm，使每个对象独占一张显卡以提升性能；并将一组较小对象分配给 LightFarm，共享 GPU资源以提高利用率。渲染包含多个对象的帧 T_{frame}所花费的总时间可表示如下：

$$T_{frame} = \max_{i \subset O}(T_i) + \sum_{i \subset O} \frac{D_i}{BW} + C \tag{9}$$

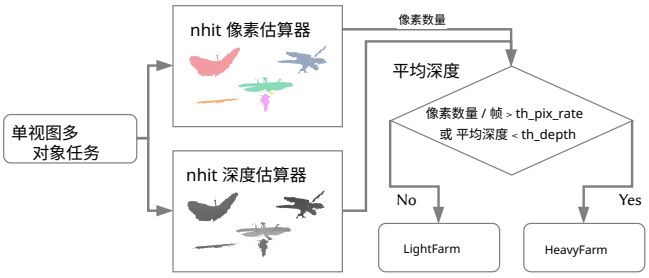

图 8. **多对象调度** 该流程图描绘了多个对象的调度过程。收到单视图多对象任务后，调度器中的两个估算器会计算*nhit*的像素数量（*n*_像素数量）和平均深度（平均_深度），然后将这些数据传递给决策模块。需要更高像素_速率或更低深度的任务，将在 HeavyFarm 中分配一张或多张专用显卡。其余任务则在 LightFarm 中运行。

其中，T_i 为渲染时间，D_i 为 NGP-NOLF 实例 i 的结果数据大小，该大小与图像分辨率有关；BW为主节点的带宽限制；C 为整合所有渲染结果所需的时间；O 为渲染实体的集合。调度器的目标是在保持预设帧分辨率（表示为 D_i）的同时，降低 T_{frame}。调度器应尽可能减少 O 中的实例数量，并在渲染农场中选择合适的子集。我们使用预先设定的网格来估算最终渲染屏幕上的像素数量和深度，以预测相应 *nhit* 的数量。借助这些指标，我们可以确定渲染任务是否必要，以及目标渲染农场。图 8 展示了调度器的基本原理。

我们使用 EGL 生成粗略的先验场景，并将其离屏渲染为*nhit*估算器；该估算器运行速度极快，因此对渲染时间的影响可以忽略不计。我们采用 InfiniBand 和支持 CUDA 的 MPI 进行跨节点通信，并利用 Nvidia Advanced GPU 上的 GPU Direct RDMA 优化，加快渲染结果的读回，从而降低跨节点开销。

6.2 多用户调度

对于多用户场景，最简单的做法是以轮询方式渲染每位用户的任务，以平衡公平性和平均响应时间。但如果用户从相邻角度观察同一物体，由于这些角度共享相当数量的渲染光线，这种做法会造成重复渲染。此外，轮询的公平性策略在满足每位用户的渲染响应时间方面存在局限，导致部分用户的每秒帧数（FPS）无法达到要求。我们提出一种多用户调度方案，以提高渲染效率并满足用户对每秒帧数的需求。

6.2.1 提高渲染效率。我们首先讨论如何优化为观看同一物体的多个用户进行渲染的响应时间（T）。我们有

$$T = \left((kT_{M_S} + T_{N_S}) \times n + \sum_{i=0}^{n} (kT_{M_i} + T_{N_i}) \right) \tag{10}$$

算法1： NEPHELE 调度策略

```
当 渲染农场可用时 执行
    对于 u ∈ 用户 执行
        u.已加载_任务 ← 已加载的任务数;
        u.目标_时间 ← T_last帧 + T_wait;
    end
    sort(users，按 loaded_tasks和 target_time);
    // 首先解决饥饿问题
    starved_users ← 用户其target_time ≤ current_time;
    当 starved_users ≠ ∅ 执行
        拒绝来自新客户端的请求;
        对于 u ∈ 饥饿用户 执行
            在渲染农场上渲染 u 的所有任务;
            更新 u的渲染时间;
        end
        允许来自新客户端的请求;
    end
    // 渲染直至渲染农场满载
    光线数 ← 渲染农场支持的最大光线数;
    对于 u ∈ 用户执行
        如果光线不足以完成渲染 u 则中断;
        对于 t ∈ 共享任务 u 执行
            在 RenderFarm 上渲染 t;
            为所有用户更新 t的渲染时间;
            光线 ← 光线 − 的光线 t;
        end
        针对 t ∈ 专用任务 u 执行
            在渲染农场上渲染 t;
            更新 u的渲染时间;
            光线 ← 光线 − 的光线 t;
        end
    end
端
```

其中， k 是常数系数， T_M 是网格渲染所需的时间， T_N 是 i-NOLF 渲染所需的时间， n 是请求渲染服务的用户数量。请注意， S 指相邻角度和全局视图之间可共享的光线，而 i 是渲染特定视角所需的光线。因此， $S \times n$ 和 $\sum i$ 的并集就是提交的渲染作业总数。由于效率 E 是 T 的倒数，我们可以发现，通过最大限度地增加可共享光线的数量，充分利用更多可共享光线能够显著提高渲染效率。

主节点中的任务分配器将可共享光线视为重型任务，并将它们全部放置到 HeavyFarm 中，以获得更好的性能。特定视角所需的光线会分配给 LightFarm 中可用的任意显卡。在这种情况下，不同角度所需的多条光线可以并行渲染。这样，公式 10 可表示为：

$$T = \left(\left(kT_{M_S} + T_{N_S} \right) + \max_{i=0}^{n} \left(kT_{M_i} + T_{N_i} \right) \right) \tag{11}$$

任务分配器还维护一个历史表，用于缓存可重复使用的任务，并指导渲染农场离线预先渲染可共享光线

在这种情况下，公式 11 还可以进一步优化为：

$$T = \max_{i \subset n} \left(kT_{M_i} + T_{N_i} \right) \tag{12}$$

6.2.2 满足用户的 *FPS* 挑战。除了渲染效率外，我们的系统还提供尽力而为的服务，通过避免渲染过多用户来实现用户请求的 FPS，如算法1所示。任务分配器为每位用户分配一个等待时间 $w_i i$，并且我们有：

$$w_i = \frac{1}{\mathrm{FPS}_i} - t_i \tag{13}$$

其中， FPS_i 是目标 FPS， t_i 是以秒为单位的作业完成时间移动平均值。 w_i 会在渲染任务完成时更新。如果某个任务在上一帧之后的 w_i 内尚未开始，则该任务处于饥饿状态。当存在饥饿任务时，任务分配器将进入饥饿状态，并拒绝新客户端的请求，直到所有饥饿任务都得到处理。

7 应用与演示

借助这样的神经云渲染平台，我们进一步通过一系列应用展示云端辐射渲染能力，包括云端 VR/AR 渲染、多个用户在移动设备上共享 NeRF 资产（如图9所示），以及将 NeRF 资产自由组合成新场景（如图10所示）。

VR 应用。头戴式显示器（HMD）为用户带来沉浸式虚拟现实体验。我们的解决方案包括前端和后端，可让用户直接在浏览器中交互式地享受 VR 流媒体传输。前端采用 Web 框架 A-Frame，支持所有兼容 WebXR 的浏览器。前端注册为 A-Frame 系统后，在检测到 HMD 时会与后端建立 WebRTC 连接。连接建立后，我们以至少 20 毫秒的间隔，持续从 WebXR 摄像机获取用户的视图和姿态。Three.js 将视图和姿态转换为与 Instant-NOLF 中相同的矩阵，然后通过 RTCDataChannel 发送给后端。后端根据客户端数据渲染内容并生成立体帧，再通过 WebRTC 将其传输给客户端。前端会对视频进行解码，并通过更改 A-Frame视频基本体的网格元素中的 uv 属性，将视频拆分为两个视图。

AR 应用。我们的基于流式传输的方案，让用户能够通过无线网络在 iPad 或手机上交互式体验高保真、照片级真实感的增强现实，并将一盆花投影在眼前。用户可以打破现实世界与虚拟世界之间的界限，以沉浸式方式从任意角度欣赏这些花朵。我们在 AR 场景下的实现细节如下：移动设备使用 ARKit 的 ARCamera 持续获取相机的内参矩阵和变换矩阵，后者定义了相机在世界坐标中的旋转和平移。当相机检测到平面时，我们还会获取一个 ARPlaneAnchor，它记录了平面在锚点坐标系中的中心和范围。我们通过 WebRTC 的 RTCDataChannel，每 20 毫秒将这些矩阵和 ARPlaneAnchor 直接从移动设备传输到云服务器。云服务器使用纯

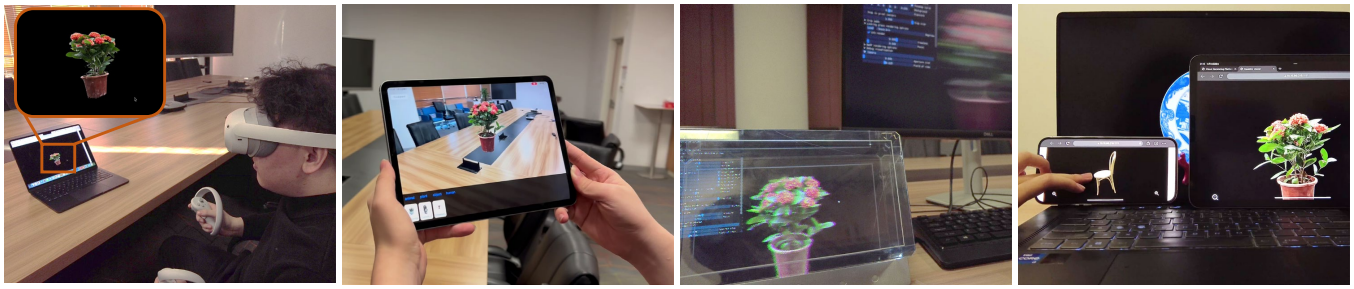

图 9. **NEPELE 应用示意图，包括 VR 应用、AR 应用、在 Looking Glass 全息显示屏上的渲染，以及移动设备。**



图10. **NEPELE 多用户、多物体场景应用示例，包括乐高星球大战（红色高亮）、昆虫大战（蓝色高亮）和中国生肖（绿色高亮）。**

绿色背景，根据来自移动设备的参数渲染内容，并使用低延迟预设的 NVENC 编码器将渲染帧编码为比特流。随后，通过 WebRTC 以低延迟将比特流传输到移动设备。收到比特流后，移动设备会解码视频帧，并使用色度键抠像法将解码帧的背景设为透明。最后，我们将这些帧叠加到 ARSCNView 的相机画面上，融合现实世界与虚拟世界，打造沉浸式体验。

Looking Glass 显示屏。Looking Glass 是一种 3D 全息显示器，需要 45 个水平连续的相机位姿来渲染 Looking Glass 场景。相机的目标是模拟三维场景处于真实物理空间时人眼所看到的画面。最简单的理解方式是，设想 Looking Glass 底部的屏幕是一块窗玻璃：一个平坦的矩形入口，我们透过它观看三维物体。现有方法很难同时渲染 45 个视图。我们在 i-NGP 上尝试了这一方法，结果发现，采用我们的方法后，i-NGP 的渲染速度变为 2–5 每秒帧数。显然，45 个视图之间有许多冗余光线可以重复利用。我们的方法非常适合这种渲染模式，渲染速度可大幅提升，最高可达 10-20+ 每秒帧数。

移动设备上的云渲染。 我们提出的 NEPELE 平台还支持各种移动设备，使 NeRF 技术不再受限于高性能计算设备，让所有人都能使用神经辐射场。我们在计算集群上运行 NEPELE 平台，用户连接到我们的云渲染平台服务后，即可通过手机、平板电脑和笔记本电脑上的 Web 查看器体验 NeRF 场景。根据用户的网络带宽，单用户在移动设备上的云渲染速度可达 25+ 每秒帧数。无需下载额外的模型数据，即可使用移动设备通过我们的云渲染平台体验所有结果。

面向多用户场景的云渲染。 传统的 NeRF 方法存在的问题在于，它们独立运行于一台

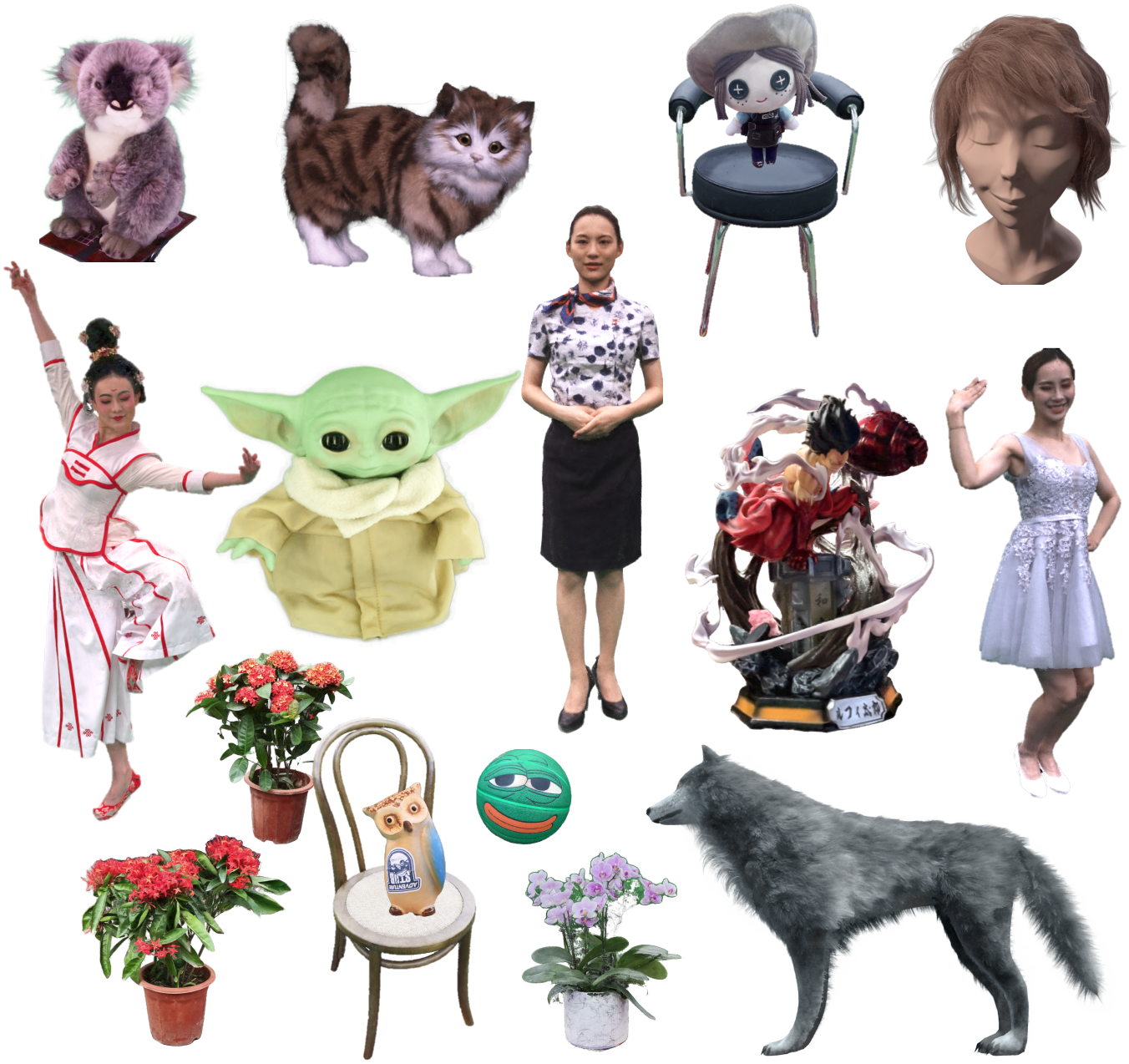

图11。**我们的合成结果集锦**。我们的方法能够很好地泛化到各种物体，包括具有不透明表面的物体、几何结构复杂的物体，以及边界模糊的物体，例如人类、玩具、动物、植物、家具和雕塑。

本地计算设备上。因此，它们不具备让多人共享 NeRF 场景表示以及组合 NeRF 场景的功能。我们提出的 NEPELE 平台提供了大量 NeRF 场景，不仅支持多个用户体验某个 NeRF 场景，还允许将 NeRF 场景自由组合成新场景。NEPELE 得益于由远程云端，以及专门设计的 Instant-NOLF 和光线级调度算法，使得即使有多个用户，也能实时渲染多个 NeRF 组合场景。

8 实验

在本节中，我们在各种具有挑战性的场景下评估 i-NOLF。首先介绍用于评估我们方法的数据集，以及与我们的方法进行比较的当前最先进（SOTA）神经渲染方法。然后，我们从渲染质量和渲染速度两方面，将模型与基线模型进行比较。结果表明，在保持出色渲染质量且与基线方法相当的同时，我们的方法能够实现远高于基线方法的渲染速度；与 i-NGP 相比，在渲染负载较高的场景（例如 4K 分辨率渲染）下尤其如此。我们还进行了消融实验，以验证方法中各个组件的有效性。此外，我们深入分析了速度-质量权衡，为将模型部署到云端提供关键信息。最后一小节将讨论系统的局限性并作进一步探讨。

数据集。 为与其他方法进行公平比较，我们采用常用的基准数据集 Synthetic-NeRF [Mildenhall 等人 2021] 。此外，我们还拍摄了各种真实物体，以验证我们的方法在不同场景下的泛化能力。具体而言，我们使用配备 80 台摄像机的绿幕穹顶，拍摄了若干身着不同服装、摆出复杂姿势的人类；同时使用手机为一些常见的不同材质和形状的物体拍摄 100 至 200 张图像，例如盆栽植物、雕塑、家具和玩偶。我们利用现成的背景抠像方法 [Lin 等人 2021] 获取前景 Alpha 蒙版。我们均匀选取 10% 的图像用于测试，其余用于训练。最后，我们还采用 ConvNeRF [Luo 等人2021] 中的模糊物体数据集，展示我们的方法处理几何结构极其复杂物体的能力，例如“狼”和“毛发”。图 11 展示了部分具有代表性的渲染结果。

基线方法。 我们采用原始 NeRF [Mildenhall 等人 2021] 作为基线模型。此外，我们还将本方法与近期几种快速神经渲染技术进行比较，包括 DVGO [Sun 等人 2022], TensoRF [Chen 等, 2022b], , 以及利用显式加速数据结构和定制 CUDA 代码的方法，例如 PlenOctrees [Yu 等人 2021],PlenOxels [Sara Fridovich-Keil 和 Alex Yu 等人 2022] 以及 i-NGP [Müller 等人 2022]。我们还与 MobileNeRF[Chen 等, 2022a], 进行比较，该方法同样旨在移动设备上实现辐射度渲染，但采用了不同的方式：使用显式网格和纹理，通过传统 OpenGL 流程进行本地渲染。

8.1 渲染质量比较

在自由视角渲染方面，我们将我们的方法与基线方法进行了比较。为确保公平性，我们在相同的原始分辨率下，对基线模型和我们的方法均优化了 20 万次迭代。图1? 图12 展示了在“drums”和“wolf”场景中与基线方法进行的定性比较。不出所料，MobileNeRF 未能重建鼓的半透明表面，因为它采用二值不透明度来避免对多边形进行排序。此外，带纹理网格的能力有限，导致其无法建模镜面反射效果；而其他方法（例如 PlenOxels）则倾向于合成噪声图像。相比之下，我们的方法在恢复高光方面表现更好。对于模糊物体，我们的 i-NOLF 能够恢复清晰的几何结构和外观细节，尤其是在边界区域，这对于毛茸茸的动物至关重要，例如.....相比之下，基线方法生成的毛发细节较为模糊。具体而言，i-NGP 会产生明显的重影效应，而 MobileNeRF 受网格限制，无法建模如此复杂的几何结构。在定量比较中，我们采用峰值信噪比（**PSNR**）、结构相似性（**SSIM**）和学习感知图像块相似度（**LPIPS**）[Zhang 等人 2018] 作为指标，评估我们模型的渲染质量。我们在表2中报告了每种方法在各个数据集上的平均指标。我们的方法优于大多数基线方法，证明了我们的方法能够很好地泛化到各种类型的物体。

8.2 渲染速度比较

我们在此进一步评估 i-NOLF 的渲染速度，这对于满足云端场景中的交互需求至关重要。我们报告了场景“Lego”上的模型大小和渲染速度，并将相较于原始 NeRF 提升的峰值信噪比和结构相似性作为参考一并报告。如表 3 所示，我们的方法不仅实现了超快的渲染速度，也取得了更好的渲染质量。具体而言，TensoRF 的渲染质量最佳，但渲染速度较慢，仅为每秒 2.04 帧；而我们的方法可达到 400+ FPS。PlenOctree 的渲染速度相近，但渲染质量有所损失。需要注意的是，尽管 MobileNeRF 的速度达到 1000+ fps，但它是一种本地渲染器，需要多个网格和纹理，且训练时间很长、预处理流程复杂，同时渲染质量也不如 NeRF。此外，网格和纹理体积较大，移动设备从服务器加载数据后才能开始渲染，因此等待时间较长；而我们的云端解决方案能够自然地实现无缝的用户体验。如图 13 所示，我们的方法在渲染质量、速度与模型大小之间实现了最佳权衡。

此外，我们通过拉近相机来评估 i-NGP 和我们的 i-NOLF 在渲染重型任务时的速度。我们在单个 Nvidia GeForce RTX4090 图形处理器上以 4K 分辨率渲染场景“Lego”。如图14所示，当相机距离较远时，i-NGP 的渲染速度与 i-NOLF 相近，但 i-NGP 的渲染时间会以指数级增长，变得不可接受。相反，由于渲染一个像素只需一次网络查询，i-NOLF 的渲染时间几乎呈线性增长。最后，当渲染对象填满整个渲染屏幕时，i-NGP 的每秒帧数仅为 2.63 FPS，表现较差，而我们的 i-NOLF 仍可达到约 32.28 FPS。这表明我们的 i-NOLF 具有出色的渲染效率。

8.3 消融实验

我们还在 NeRF 合成数据集上对我们的方法进行了消融研究，如表4所示。我们训练了一个模型（“无命中点”），其粗略代理几何体略大于真实物体，结果性能显著下降，这表明命中点估计策略有助于近似神经不透明度光场。移除视角相关的不透明度输出（“无不透明度”）会严重降低渲染指标，因为

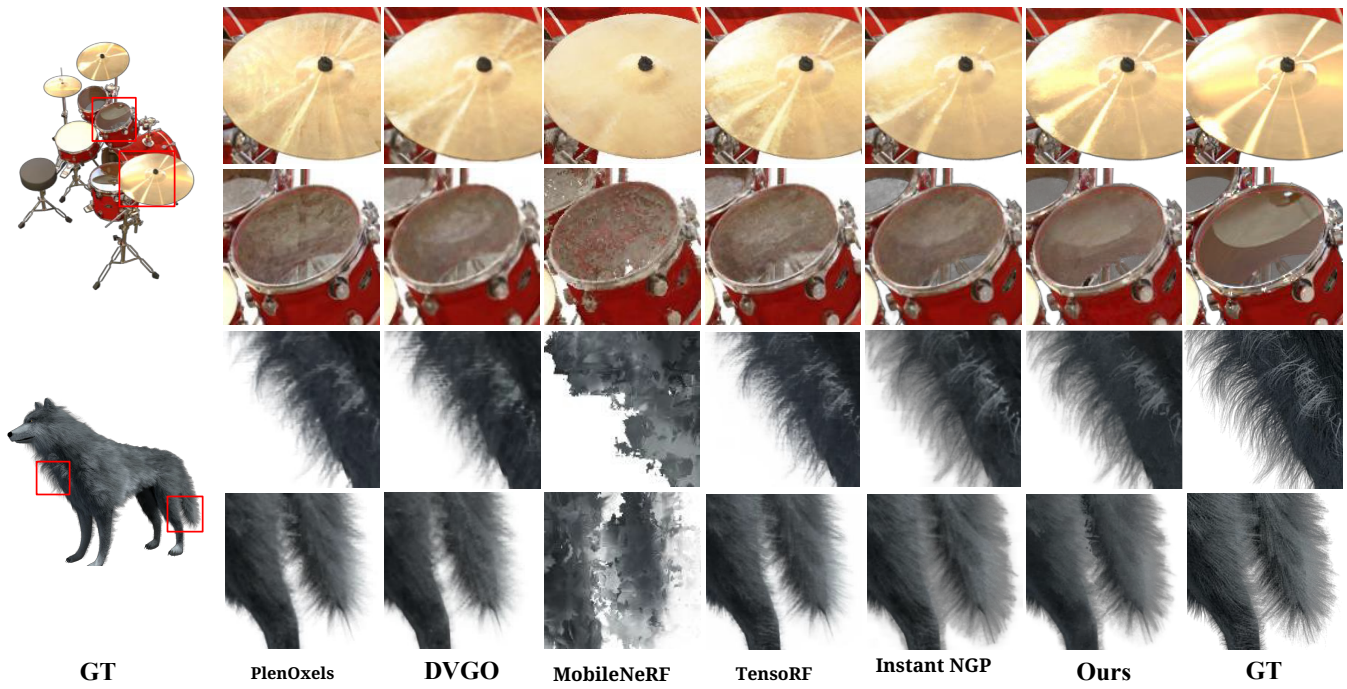

图1 2. **自由视点合成的定性比较**。我们的方法能够更好地恢复镜面反射效果和毛发细节 ils.

表2。**各数据集上的定量比较**。我们报告了基线方法和我们方法的平均指标。总体而言，我们的方法实现了最佳的渲染质量。

方法	Synthetic-NeRF			真实采集			模糊物体		
	峰值信噪比 ↑	结构相似性 ↑	学习感知图像块相似度 ↓	峰值信噪比 ↑	结构相似性 ↑	学习感知图像块相似度 ↓	峰值信噪比 ↑	结构相似性 ↑	学习感知图像块相似度 ↓
NeRF	31.00	0.947	0.081	27.83	0.930	0.087	31.20	0.917	0.102
PlenOctree	30.39	0.946	0.073	24.86	0.918	0.104	31.35	0.915	0.114
PlenOxels	31.74	0.958	0.049	27.30	0.935	0.102	32.04	0.930	0.100
MobileNeRF	30.89	0.947	0.062	27.02	0.902	0.108	30.82	0.911	0.110
DVGO	31.95	0.957	0.053	29.22	0.938	0.080	33.69	0.925	0.106
TensoRF	33.21	0.963	0.051	28.60	0.948	0.085	34.67	0.948	0.081
i-NGP	32.79	0.920	0.061	29.48	0.919	0.065	34.35	0.881	0.105
Ours	32.73	0.960	0.049	29.62	0.947	0.058	34.61	0.928	0.073

表 3. **渲染速度的定量比较**。我们报告了 NeRF 合成数据集上的平均指标，以及在“Lego”场景中以 800x800 分辨率测得的模型大小和渲染速度。我们的方法在保持高渲染质量的同时，实现了显著更快的速度。

方法	Δ峰值信噪比↑	Δ结构相似性↑	大小（MB）↓	时间（ms）↓	每秒帧数↑	加速比↑
NeRF	0.00	0.000	13.6	10417.28	0.09	1.00
PlenOctree	-0.61	-0.001	446.9	2.11	473.93	5265.89
PlenOxels	0.74	0.011	812.3	56.39	17.73	197.00
MobileNeRF	-0.11	0.000	176.1	0.83	1198.86	13320.67
DVGO	0.95	0.010	652.8	63.49	15.75	175.00
TensoRF	2.21	0.016	68.8	488.23	2.04	22.67
i-NGP	1.79	-0.027	28.7	6.84	146.18	1624.22
Ours	1.73	0.013	48.8	2.07	482.30	5358.89

缓存密度场无法提供与真实物体完全相同的几何。此外，不使用粗略不透明度输入而直接预测不透明度（“无不透明度细化”）也会降低渲染性能。最后，将镜面反射色调替换为一个常数值（例如 0.5）会使性能略微下降（“无色调”）；而不包含漫反射分量的模型（“无漫反射颜色”）则说明了解耦漫反射和镜面反射分量的优势。图15展示了不同模型的视觉细节。没有命中点估计时，由于浅层多层感知机的能力有限，学习得到的不透明度光场相当嘈杂。若不预测视角相关不透明度，模型便无法重建“ficus”等几何结构复杂的物体，而不透明度细化策略则进一步有助于恢复精细细节。

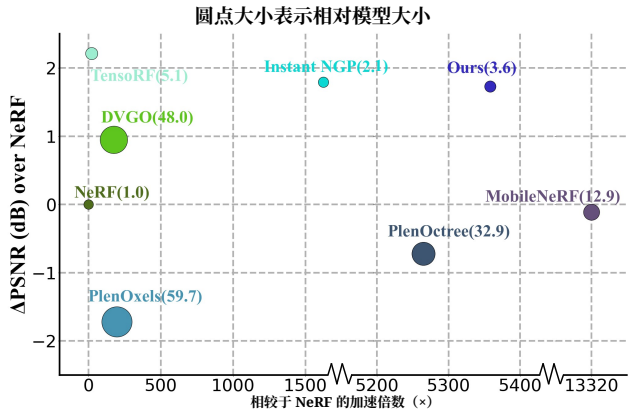

图13. **加速倍数-峰值信噪比-模型大小对比**。与其他快速神经渲染方法相比，我们的方法在加速倍数、峰值信噪比和模型大小之间实现了更优的权衡。请注意，圆点大小表示相对模型大小；我们的方法兼具快速渲染、高质量渲染和小模型大小。

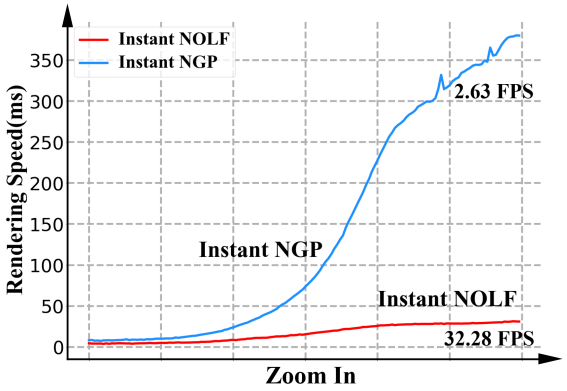

图14. **速度-距离对比**。我们比较了摄像机逐渐拉近时 i-NOLF 与 i-NGP 的渲染速度。当查看器逐渐靠近时，i-NGP 的渲染时间呈指数级增长，而 i-NOLF 仍能保持出色的速度，即使在近距离观察位置也是如此。

8.4 速度-质量权衡分析

虽然我们的 i-NOLF 实现了更快的渲染速度，但更重要的是分析将模型部署到云端时的速度与质量权衡。如表5所示，我们在单个 Nvidia GeForce RTX4090 图形处理器上，定量分析了各个组件对速度、模型大小和性能的影响。我们报告了在完整 Synthetic-NeRF 数据集上测得的渲染质量平均指标。对于渲染速度，我们计算了在“lego”场景中以 4K 分辨率渲染 200 个随机视图的平均运行时间，以模拟云端场景中的高并发渲染工作负载。具体而言，我们还报告了光线步进过程的运行时间；在该过程中，查询网络占用了大部分时间。

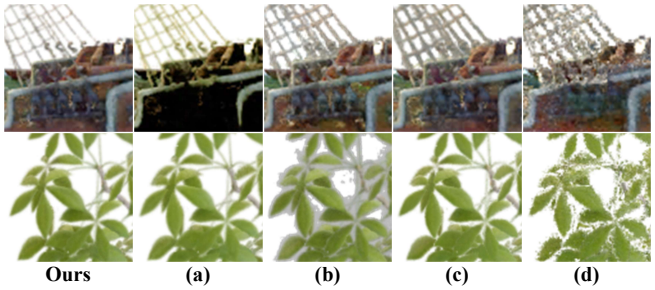

图15。**定性消融研究**。我们在具有挑战性的“ship”场景和“ficus”场景部分展示了各个组件的有效性：(a) 不含漫反射分量。(b) 不含视角相关不透明度。(c) 不含不透明度细化。(d) 不含命中点估计。

表4。**消融研究**。我们在 Syhthetic-NeRF 数据集上评估了模型的各个组件。

模型	峰值信噪比↑	结构相似性↑	学习感知图像块相似度↓
我们的方法（不含命中点）	28.48	0.924	0.102
我们的方法（不含不透明度）	23.38	0.919	0.088
我们的方法（不含不透明度细化）	32.48	0.960	0.052
我们的方法（不含色调）	32.65	0.960	0.051
我们的方法（不含漫反射颜色）	32.66	0.959	0.052
Ours, full	32.73	0.960	0.049

在表5中，首先，密度纹理（记为“dt”）使渲染帧率提高到基础模型的五至六倍；基础模型通过查询密度网络来估算命中点。使用多层感知机近似漫反射分量（“漫反射多层感知机”）可提升渲染性能，但帧率会降低 7，模型大小也会增至两倍。另一方面，采用漫反射纹理（“dift”）几乎消除了查询漫反射网络的开销，但质量略有下降。最后，我们使用完美空间哈希编码（PSH）取代多分辨率哈希编码，使光线步进时间进一步减少约 20%，总渲染时间减少 15%。提高 PSH 编码的分辨率（例如从 128^3 提高到 512^3）可略微提升渲染质量，而运行时间几乎不变。请注意，由于只需查询网络一次，我们的模型整体光线步进时间至少缩短了 90%，因此，渲染图像所需的其余时间（即初始化时分配内存）成为进一步优化的瓶颈。我们还报告了未压缩的密度纹理和漫反射纹理所需的额外运行时内存；值得注意的是，在多个用户共享同一场景纹理的云端场景中，这完全可以接受。

8.5 局限性与讨论

我们已经展示了 NEPHELE 云平台的一项强大能力：将神经辐射渲染的算力普惠给所有人，使其无需依赖高端设备，并实现一系列前所未有的效果，例如构建由许多 NeRF 对象组成的可供多用户共享的复杂场景。然而，作为首次尝试，

表5。**质量-速度权衡分析**。我们在 Synthetic-NeRF 数据集上评估各模型的渲染质量。对于渲染速度，我们报告以 4K 分辨率渲染场景“Lego”时的平均光线步进/总渲染时间（单位：毫秒）和帧率。我们还报告了模型大小以及缓存所需的额外运行时内存（单位：MB），这些对于部署我们的模型至关重要。

模型	峰值信噪比↑	结构相似性↑	大小↓	额外内存↓	时间↓	FPS↑
i-NGP	32.79	0.920	28.7	0	86.77/96.37	10.38
我们的基础版	-	-	24.4	0	97.94/107.54	9.30
我们的方法，dt	32.66	0.959	24.4	1038	11.17/20.35	49.14
Ours, dt, 漫反射多层感知机	32.73	0.960	48.8	1038	14.58/23.77	42.07
Ours, dt, dift	32.58	0.960	48.8	1435	11.31/20.60	48.54
我们的方法，dt，dift，PSH-128	32.02	0.957	30.3	1435	9.07/18.38	54.41
我们的方法，dt，dift，PSH-256	32.23	0.956	53.3	1435	9.33/18.56	53.87
我们的方法，dt，dift，PSH-512	32.36	0.954	139.6	1435	9.36/18.57	53.85

将神经辐射渲染引入云端以支持多用户使用，我们的 NEPHELE 系统仍存在以下一些局限性。

首先，从算法层面来看，所提出的 i-NOLF 依赖于 i-NGP 预训练得到的密度场，因此无法有效处理 i-NGP 无法重建出有意义几何结构的情况，例如光滑表面。将近期优秀工作 RefNeRF [Verbin 等人 2022] 与 i-NGP 相结合，有望解决这一问题。此外，为实现超快速渲染，所采用的命中点估计经验策略对每条光线仅生成一个点。然而，对于多层物体，该策略效果不佳，例如由前方的半透明表面和后方的不透明表面构成的物体。未来，我们计划开发一种更全面的策略，类似于传统渲染管线中的半透明材质渲染技术。此外，目前的 i-NOLF 只能在捕获物体时所处的光照环境中重建该物体。将近期的基于表面的神经重照明技术扩展到不透明度光场，以实现更丰富的视觉效果，是一种很有前景且自然的做法。最后，尽管我们已将 i-NOLF 部署在云端，但实时渲染仍需要高端 GPU。进一步探索开发类似于 [Chen 等，2022a] 的本地渲染器，使其适用于移动设备，将是一项有趣且有意义的工作。

从系统层面来看，我们的 NEPHELE 系统仅将任务分为轻型和重型两类并进行调度，这可能导致显卡出现外部碎片。我们计划采用多级分层分类器对任务进行分类，使其适配显卡的计算能力。此外，我们计划扩展 NEPHELE，使其兼容异构图形处理器资源。我们还尝试构建一个配备高性能存储系统的 RayFarm，以加快对预渲染光线的访问。借助 RayFarm，渲染农场能够消除查找已渲染光线以及将新光线存入存储池的开销。

9 结论

我们提出了 NEPHELE，这是一个用于高度逼真云端辐射渲染的神经平台。与现有的神经辐射方法截然不同，我们的 NEPHELE 通过组合多个远程 GPU，提供更强大的渲染能力；同时支持多人同时查看同一 NeRF 场景，从而促进协作。为在基于云端的场景中实现具备可扩展性的轻量级实时神经表示，我们引入了 i-NOLF，采用类似于 i-NGP 的不透明度光场。我们的 i-NOLF 进一步结合带有几何代理的 Lumigraph，以一条光线一次查询的方式实现超高效的神经辐射渲染，并支持快速光线查询。随后，我们为局部不透明度光球采用轻量级多层感知机，实现高质量的视点相关渲染。不同于 i-NGP 中使用多分辨率哈希的方法，我们采用完美空间哈希来显著提升缓存一致性，并通过单分辨率编码设计进一步减少缓存流量。我们还针对 i-NOLF 表示设计了任务调度器，并采用光线级调度设计，以维持渲染作业的韧性。为此，我们构建了一个完整的云平台，其中包含多个协同模块，即渲染农场、任务分配器、帧合成器以及详细的流媒体策略。在多个数据集上的广泛定性和定量结果表明，i-NOLF 对于基于云端的辐射度渲染场景十分有效。我们进一步展示了 NEPHELE 的多种前所未有的应用，从虚拟现实/增强现实中的云端辐射渲染，到体验、分享和组装各种 NeRF 资产。凭借这些独特优势，我们相信，该方法通过一种新颖的基于云端的范式，为普及辐射度渲染的使用铺平了道路，并能够在虚拟现实/增强现实领域支持大量潜在应用，包括娱乐和沉浸式体验。

参考文献

Yixin Bao、Yanghua Peng、Chuan Wu 和 Zongpeng Li。2018年。《分布式机器学习集群中的在线作业调度》。载于*IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*（美国夏威夷州檀香山）。IEEE Press，495–503。https: // doi.org/10.1109/INFOCOM.2018.8486422 Jonathan T Barron、Ben Mildenhall、Matthew Tancik、Peter Hedman、Ricardo Martin- Brualla 和 Pratul P Sri nivasan。2021年。《Mip-nerf：用于抗锯齿神经辐射场的多尺度表示》。载于*IEEE/CVF* 国际计算机视觉会议论文集。5855–5864。Zhengda Bian、Shenggui Li、Wei Wang 和 Yang You。2021年。《用于调度 GPU 集群深度学习工作负载的在线进化批量大小编排》。载于高性能计算、网络、存储与分析国际会议论文集（密苏里州圣路易斯）*(SC '21)*。美国计算机协会，美国纽约州纽约市，文章 100，共 15 页。https://doi.org/ 10.1145/3458817.3480859 Mark Boss、Raphael Braun、Varun Jampani、Jonathan T. Barron、Ce Liu 和 Hendrik P.A. Lensch。2021年。《NeRD：从图像集合中进行神经反射分解》。载于*IEEE* 国际计算机视觉会议（*ICCV*）。Chris Buehler、Michael Bosse、Leonard McMillan、Steven Gortler 和 Michael Cohen。2001年。《非结构化光场渲染》。载于第 *28* 届计算机图形学与交互技术年度会议论文集。425–432。Anpei Chen、Minye Wu、Yingliang Zhang、Nianyi Li、Jie Lu、Shenghua Gao 和 Jingyi Yu。2018年。《深度表面光场》。*ACM* 计算机图形学与交互技术论文集1, 1，文章 14（2018年7月），共 17 页。https://doi.org/10.1145/3203192 Anpei Chen、Zexiang Xu、Andreas Geiger、Jingyi Yu 和 Hao Su。2022b。《TensoRF：张量辐射场》。*arXiv* 预印本 *arXiv:2203.09517*（2022）。Anpei Chen、Zexiang Xu、Fuqiang Zhao、Xiaoshuai Zhang、Fanbo Xiang、Jingyi Yu 和 Hao Su。2021年。《Mvsnerf：基于多视图立体的快速通用辐射场重建》。载于*IEEE/CVF* 国际计算机视觉会议论文集。14124–14133。Zhiqin Chen、Thomas Funkhouser、Peter Hedman 和 Andrea Tagliasacchi。2022a。《Mobilenerf：利用多边形光栅化流水线，在移动架构上高效渲染神经场》。*arXiv* 预印本 *arXiv:2208.00277*（2022）。Andrea Di Domenico、Gianluca Perna、Martino Trevisan、Luca Vassio 和 Danilo Giordano。2021年。《云游戏网络分析：Stadia、GeForce Now 和 PSNow》。*Network* 1, 3（2021），247–260。https://doi.org/10.3390/network1030015 Edward A. Fox、Lenwood S. Heath、Qi Fan Chen 和 Amjad M. Daoud。1992年。《大型数据库的实用最小完美哈希函数》。*Commun. ACM* 35, 1（1992年1月），105–121。https://doi.org/10.1145/129617.129623

Juncheng Gu、Mosharaf Chowdhury、Kang G. Shin、Yibo Zhu、Myeongjae Jeon、Junjie Qian、Hongqiang Liu 和 Chuanxiong Guo。2019年。《Tiresias：用于分布式深度学习的 GPU 集群管理器》。载于第 *16* 届 *USENIX* 网络系统设计与实现会议论文集（美国马萨诸塞州波士顿）*(NSDI'19)*。USENIX 协会，美国，485–500。Michelle Guo、Alireza Fathi、Jiajun Wu 和 Thomas Funkhouser。2020年。《以对象为中心的神经场景渲染》。*arXiv* 预印本 *arXiv:2012.08503*（2020）。Van Kha Ly Ha、Rifai Chai 和 Hung T. Nguyen。2020年。《采用 WebRTC 和 360 度视野的远程临场轮椅》。《应用科学》10, 1（2020）。https: //doi.org/10.3390/app10010369 Myeongjae Jeon、Shivaram Venkataraman、Amar Phanishayee、unjie Qian、Wencong Xiao 和 Fan Yang。2019年。《面向 DNN 训练工作负载的大规模多租户 GPU 集群分析》。载于 *2019* 年 *USENIX* 年度技术会议论文集（美国华盛顿州伦顿）*(USENIX ATC '19)*。USENIX 协会，美国，947–960。Randell Jesup、Salvatore Loreto 和 Michael Tuexen。2015年。《WebRTC 数据通道》。互联网工程任务组，互联网草案，*draft-ietf-rtcweb-data-channel-13*（2015）。Yongjae Lee、Byounghyun Yoo 和 Soo-Hong Lee。2021年。《在基于 Web 的 XR 远程协作中，通过实时点云流传输共享环境物体》。载于第 *26* 届国际 *3D Web* 技术会议。1–9。Sylvain Lefebvre 和 Hugues Hoppe。2006年。《完美空间哈希》。*ACM* 图形学汇刊 25, 3（2006年7月），579–588。https://doi.org/10.1145/1141911.1141926 Shanchuan Lin、Andrey Ryabtsev、Soumyadip Sengupta、Brian L Curless、Steven M Seitz 和 Ira Kemelmacher-Shlizerman。2021年。《实时高分辨率背景抠像》。载于*IEEE/CVF* 计算机视觉与模式识别会议论文集。8762–8771。Lingjie Liu、Marc Habermann、Viktor Rudnev、Kripasindhu Sarkar、Jiatao Gu 和 Christian Theobalt。2021年。《神经演员：通过姿态控制对人体角色进行神经自由视点合成》。*ACM* 图形学汇刊（*ACM SIGGRAPH Asia*）（2021）。Stephen Lombardi、Tomas Simon、Gabriel Schwartz、Michael Zollhoefer、Yaser Sheikh 和 Jason Saragih。2021年。《用于高效神经渲染的体积基元混合》。*ACM* 图形学汇刊（*TOG*）40, 4（2021），1–13。H. Luo、A. Chen、Q. Zhang、B. Pang、M. Wu、L. Xu 和 J. Yu。2021年。《卷积神经不透明度辐射场》。载于*2021* 年 *IEEE* 国际计算摄影会议（*ICCP*）。IEEE 计算机学会，美国加利福尼亚州洛斯阿拉米托斯，1–12。https: //doi.org/10.1109/ICCP51581.2021.9466273 Haimin Luo、Teng Xu、Yuheng Jiang、Chenglin Zhou、Qiwei Qiu、Yingliang Zhang、Wei Yang、Lan Xu 和 Jingyi Yu。2022。《Artemis：具有外观与运动合成能力的可动神经宠物》。*ACM* 图形学汇刊 41, 4，文章 164（2022年7月），共 19 页。https://doi.org/10.1145/3528223.3530086 Blair Maclntyre 和 Trevor F Smith。2018年。《关于 WebXR 和沉浸式 Web 未来的思考》。载于*2018* 年 *IEEE* 混合与增强现实国际研讨会（*ISMAR-Adjunct*）。IEEE，338–342。Kshiteej Mahajan、Arjun Balasubramanian、Arjun Singhvi、Shivaram Venkataraman、Aditya Akella、Amar Phanishayee 和 Shuchi Chawla。2020年。《THEMIS：公平且高效的 GPU 集群调度》。载于第 *17* 届 *USENIX* 网络系统设计与实现会议论文集（美国加利福尼亚州圣克拉拉）*(NSDI'20)*。USENIX 协会，美国，289–304。Wojciech Matusik、Hanspeter Pfister、Addy Ngan、Paul Beardsley、Remo Ziegler 和 Leonard McMillan。2002年。《利用不透明度包壳进行基于图像的 3D 摄影》。*ACM* 图形学汇刊 21, 3（2002年7月），427–437。https://doi.org/10.1145/566654.566599 Ben Mildenhall、Pratul P Srinivasan、Matthew Tancik、Jonathan T Barron、Ravi Ra- mamoorthi 和 Ren Ng。2021年。《Nerf：将场景表示为神经辐射场以实现视图合成》。*Commun. ACM* 65, 1（2021），99–106。Thomas Müller、Alex Evans、Christoph Schied 和 Alexander Keller。2022。《采用多分辨率哈希编码的即时神经图形基元》。*ACM* 图形学汇刊 41, 4，文章 102（2022年7月），共 15 页。https://doi.org/10.1145/3528223. 3530127 Michael Niemeyer 和 Andreas Geiger。2021年。《Giraffe：将场景表示为合成式生成神经特征场》。载于*IEEE/CVF* 计算机视觉与模式识别会议论文集。11453–11464。Zainab Oufqir、Abdellatif El Abderrahmani 和 Khalid Satori。2020年。《用于增强现实服务的 ARKit 和 ARCore》。载于*2020* 年智能系统与计算机视觉国际会议（*ISCV*）。IEEE，1–7。Xingang Pan、Xudong Xu、Chen Change Loy、Christian Theobalt 和 Bo Dai。2021年。《用于形状精确的 3D 感知图像合成的着色引导生成式隐式模型》。载于神经信息处理系统进展（*NeurIPS*）。Roger Pantos。2022。《*HTTP* 实时流第二版》。互联网草案 draft-pantos-hls- rfc8216bis-12。互联网工程任务组。https://datatracker.ietf.org/doc/ draft-pantos-hls-rfc8216bis/12/ 进行中的工作。Abhijit Patait 和 Eric Young。2016。《使用 NVIDIA GPU 进行高性能视频编码》。载于*2016* 年 *GPU* 技术大会（*https://goo. gl/Bdjdgm*）。Sida Peng、Yuanqing Zhang、Yinghao Xu、Qianqian Wang、Qing Shuai、Hujun Bao 和 Xiaowei Zhou。2021年。《神经人体：采用结构化潜在编码的隐式神经表示，用于动态人体的新视角合成》。载于论文集 *IEEE/CVF* 计算机视觉与模式识别会议. 9054–9063. Yanghua Peng、Yixin Bao、Yangrui Chen、Chuan Wu 和 Chuanxiong Guo。2018年。Optimus：面向深度学习集群的高效动态资源调度器。载于第十三届 *EuroSys* 会议论文集（葡萄牙波尔图）（*EuroSys '18*）。美国计算机协会，纽约州纽约市，美国，文章 3，共 14 页。https://doi.org/10.1145/3190508.3190517 Albert Pumarola、Enric Corona、Gerard Pons-Moll 和 Francesc Moreno-Noguer。2020年。D-NeRF：用于动态场景的神经辐射场。载于*IEEE/CVF* 计算机视觉与模式识别会议。Sara Fridovich-Keil 和 Alex Yu、Matthew Tancik、Qinhong Chen、Benjamin Recht 和 Angjoo Kanazawa。2022。Plenoxels：无需神经网络的辐射场。载于计算机视觉与模式识别会议。Pratul P. Srinivasan、Boyang Deng、Xiuming Zhang、Matthew Tancik、Ben Mildenhall 和 Jonathan T. Barron。2021年。NeRV：用于重新照明和视图合成的神经反射与可见性场。载于计算机视觉与模式识别会议。Thomas Stockhammer。2011年。HTTP动态自适应流传输——：标准与设计原则。载于第二届*ACM*多媒体系统年会论文集（美国加利福尼亚州圣何塞）（*MMSys '11*）。美国计算机协会，纽约州纽约市，美国，133–144。https://doi.org/10.1145/1943552.1943572 George Suciu、Stefan Stefanescu、Cristian Beceanu 和 Marian Ceaparu。2020年。WebRTC在实时通信和视频会议中的作用。载于*2020*年全球物联网峰会（*GIoTS*）。1–6。https://doi.org/10.1109/GIOTS49054.2020.9119656 Cheng Sun、Min Sun 和 Hwann-Tzong Chen。2022。直接体素网格优化：辐射场重建的超快速收敛。载于*IEEE/CVF* 计算机视觉与模式识别会议论文集。5459–5469。Edgar Tretschk、Ayush Tewari、Vladislav Golyanik、Michael Zollhöfer、Christoph Lass-ner 和 Christian Theobalt。2021年。非刚性神经辐射场：从单目视频重建动态场景并合成新视角。载于*IEEE* 国际计算机视觉会议（*ICCV*）。电气与电子工程师协会。Dor Verbin、Peter Hedman、Ben Mildenhall、Todd Zickler、Jonathan T. Barron 和 Pratul P. Srinivasan。2022。Ref-NeRF：神经辐射场的结构化视角依赖外观。计算机视觉与模式识别会议（2022）。Esa Vikberg 等人。2021年。优化 WebRTC，以支持扩展现实的云端流传输。（2021年）。Daniel Vlasic、Hanspeter Pfister、Sergey Molinov、Radek Grzeszczuk 和 Wojciech Matusik。2003年。不透明度光场：利用视角相关不透明度进行表面光场的交互式渲染。载于*2003*年交互式三维图形研讨会论文集（加利福尼亚州蒙特雷）（*I3D '03*）。美国计算机协会，纽约州纽约市，美国，65–74。https://doi.org/10.1145/641480.641496 Huan Wang、Jian Ren、Zeng Huang、Kyle Olszewski、Menglei Chai、Yun Fu 和 Sergey Tulyakov。2022a。R2L：将神经辐射场蒸馏为神经光场，以实现高效的新视角合成。载于欧洲计算机视觉会议。Liao Wang、Jiakai Zhang、Xinhang Liu、Fuqiang Zhao、Yanshun Zhang、Yingliang Zhang、Minye Wu、Lan Xu 和 Jingyi Yu。2022b。用于实时动态辐射场渲染的傅里叶PlenOctrees。*arXiv* 预印本 *arXiv:2202.08614*（2022）。Qianqian Wang 、Zhicheng Wang、Kyle Genova、Pratul P Srinivasan、Howard Zhou、Jonathan T Barron、Ricardo Martin-Brualla、Noah Snavely 和 Thomas Funkhouser。2021年。IBRNet：学习多视图图像式渲染。载于*IEEE/CVF* 计算机视觉与模式识别会议论文集。4690–4699。Daniel N. Wood、Daniel I. Azuma、Ken Aldinger、Brian Curless、Tom Duchamp、David H. Salesin 和 Werner Stuetzle。2000年。用于三维摄影的表面光场。载于第*27*届计算机图形学与交互技术年会论文集（*SIGGRAPH '00*）。ACM Press/Addison-Wesley Publishing Co.，美国，287–296。https://doi.org/10.1145/344779.344925 Wencong Xiao、Romil Bhardwaj、Ramachandran Ramjee、Muthian Sivathanu、Nipun Kwatra、Zhenhua Han、Pratyush Patel、Xuan Peng、Hanyu Zhao、Quanlu Zhang、Fan Yang 和 Lidong Zhou。2018年。Gandiva：面向深度学习的内省式集群调度。载于第*13*届*USENIX*操作系统设计与实现会议论文集（美国加利福尼亚州卡尔斯巴德）（*OSDI'18*）。USENIX 协会，美国，595–610。Alex Yu、Ruilong Li、Matthew Tancik、Hao Li、Ren Ng 和 Angjoo Kanazawa。2021年。用于神经辐射场实时渲染的 Plenoctrees。载于*IEEE/CVF* 国际计算机视觉会议论文集。5752–5761。Haoyu Zhang、Logan Staf man、Andrew Or 和 Michael J. Freedman。2017年。SLAQ：面向分布式机器学习的质量驱动调度。载于*2017*年云计算研讨会论文集（加利福尼亚州圣克拉拉）（*SoCC '17*）。美国计算机协会，纽约州纽约市，美国，390–404。https: //doi.org/10.1145/3127479.3127490 Jiakai Zhang、Xinhang Liu、Xinyi Ye、Fuqia ng Zhao、Yanshun Zhang、Minye Wu、Yingliang Zhang、Lan Xu 和 Jingyi Yu。2021a。使用分层神经表示的可编辑自由视点视频。*ACM* 图形学汇刊（*TOG*）40，4（2021），1–18。Richard Zhang、Phillip Isola、Alexei A. Efros、Eli Shechtman 和 Oliver Wang。2018年。深度特征作为感知度量的非凡有效性。*arXiv* 电子预印本，文章 arXiv:1801.03924（2018 年 1 月），arXiv:1801.03924 页。https://doi.org/10.48550/arXiv.1801.03924 arXiv:1801.03924 [cs.CV]

Xiuming Zhang、Pratul P Srinivasan、Boyang Deng、Paul Debevec、William T Free-man 和 Jonathan T Barron。2021b。Nerfactor：在未知光照下对形状和反射率进行神经分解。*ACM* 图形学汇刊（*TOG*）40，6（2021），1–18。

Fuqiang Zhao、Wei Yang、Jiakai Zhang、Pei Lin、Yingliang Zhang、Jingyi Yu 和 Lan Xu。2022。HumanNeRF：从稀疏输入高效生成的人体辐射场。载于电气与电子工程师协会/*CVF* 计算机视觉与模式识别会议论文集。7743–7753。