

Instant-ngp 结合 NeRFREN 优化反射场景渲染

陈干

摘要

神经辐射场 (NeRF^[1]) 是 ECCV 2020 年的 best paper，由 Ben Mildenhall 等人发表。NeRF 是一种利用多层感知器 (MLPs) 进行隐式神经场景体绘制的新型视图合成方法，达到了最先进的视觉质量，其产生了令人印象深刻的。NeRF 虽然可以达到优异的图像重建效果，但是其最明显的缺点是训练时间过长，使用单张 3090 的显卡运行 Blender 数据集中 Lego 模型需要耗时 2 天左右。为了缩短 NeRF 训练时长，许多优秀科研工作者先后提出各类优化方法，如将模型拆分的 FastNeRF^[2]和 SqueezeNeRF^[3]，使用球谐函数的 Plenoxels^[4]等，都在一定程度上优化了 NeRF 的训练时间长问题。2022 年，Nvidia 发表了 Instant Neural Graphics Primitives with a Multiresolution Hash Encoding^[5]，简称 Instant-ngp。Instant-ngp 主要用于解决 NeRF 在对全连接神经网络进行参数化时的效率问题，其将 NeRF 以小时计的时间开销提升到秒级。该网络主要使用基于特征向量的多分辨率哈希表^[6]提升训练速度，并基于随机梯度下降执行优化。多分辨率结构有助于 GPU 并行，能够通过消除哈希冲突减少计算。

关键词：NeRF；Instant-ngp；多分辨率哈希表

1 引言

NeRF 从发表至今（2022），NeRF 在计算机视觉界引起了广泛的关注，并成为顶级计算机视觉会议的热门，如 CVPR、ICCV、ECCV 和 CVPR。Mildenhall 等人的 NeRF 原始论文被引用超过 1300 次，而且逐年增加，越来越多的研究者加入到 NeRF 研究领域，高质量 NeRF 相关论文如雨后春笋般涌现。众多研究机构、各类大学等基于 NeRF 贡献了多篇高质量论文，谷歌或谷歌联合其他大学前后发表了多篇高质量 NeRF 论文，如扛锯齿的 Mip-NeRF^[7]及其无界场景扩展 Mip-NeRF360^[8]，用于城市室外环境的 Urban Radiance Fields^[9]，可大规模重建旧金山整个社区的 Block-NeRF^[10]等。香港中文大学、德国马普研究所和新加坡南洋理工等的学者提出使用城市规模的 CityNeRF^[11]，展现从卫星级图像到街景级图像的三维场景，腾讯 AI Lab 先后提出适用旅游图像重建三维场景的 Ha-NeRF^[12]，弥补了谷歌在 NeRF-W^{nerf-w}关于场景分解的缺陷；高动态范围神经辐射场——HDR NeRF^[13]，实现 HDR 成像与神经辐射场的结合；适用于复杂反射场景的高质量三维场景——NeRFReN^[14]等。随着各类高保真、室外大场景、动态场景等的挖掘，NeRF 训练速度慢的问题得到了重视，FastNeRF、SqueezeNeRF、Plenoxels^[4]和 pointNeRF^[15]等相继被提出，虽然都在一定程度上改进了 NeRF 训练速度，但是还远远不够。到了 2022 年，Nvidia 在自家显卡发布会上公布了 Instant-ngp，这是 NeRF 历史上里程碑式的贡献，其 5s 训练一个 NeRF 的效果震惊了业界。这为 NeRF 训练解决了速度慢的卡脖子问题，为广大科研人员提供了一条可行的道路。训练速度问题是模型训练过程中不可避免的，在以后的工作中，在 NeRF 的发展历程上，Instant-ngp 所提供的高效训练方法将发挥特别的作用，为此，本文着力于研究 Instant-ngp。

2 相关工作

神经辐射场 (NeRF) 是一种新型的隐式场景视图合成方法，在计算机视觉领域掀起了一场风暴。作为一种新颖的视图合成和三维重建方法，NeRF 模型在机器人、城市地图、自主导航、虚拟现实/增强现实等领域都有广泛的应用。自 Mildenhall 等人的原始论文以来，已有 250 多本预印本出版，其中 100 多本最终被一级计算机视觉会议接受。

2.1 基于隐式场表达的三维重建

nerf 在复杂场景中实现了高度逼真的视图合成，受到了业界的广泛关注。在其基本形式中，NeRF 模型将三维场景表示为由神经网络近似的辐射场。亮度场描述了场景中每个点和每个观看方向的颜色和体积密度：

$$F(x, \theta, \phi) \rightarrow (c, \sigma) \quad (1)$$

其中 $x = (x, y, z)$ 为世界坐标， (θ, ϕ) 为方位角和极视角， $c = (r, g, b)$ 为颜色， σ 为体积密度。这个 5D 函数近似于一个或多个多层感知器 (MLP)，有时表示为 F 。两个视角 (θ, ϕ) 通常用 $d = (d_x, d_y, d_z)$ 表示，这是一个三维笛卡尔单位向量。这种神经网络表示通过限制对 σ 体积密度 (即场景内容) 的预测与观看方向无关来约束多视图一致性，而颜色 c 被允许同时依赖于观看方向和场景内坐标。在 NeRF 模型中，这是通过将 MLP 设计为两个阶段来实现的。

第一阶段以 x 为输入， σ 为输出，高维特征向量 (原文为 256)。在第二阶段，特征向量与观察方向 d 连接，并传递给一个额外的 MLP，输出 c 。

在给定体积密度和颜色函数的情况下，使用体绘制得到任意相机射线 $r(t) = o + td$ ，使用相机位置 o 和观看方向 d 时的颜色 $C(r)$ 。

$$\int_{t_1}^{t_2} T(t) \cdot \sigma(r(t)) \cdot c(r(t), d) \cdot dt \quad (2)$$

其中， $T(t)$ 为累计透射率，表示射线从 t_1 传到 t_2 而不被拦截的概率，表示为：

$$T(t) = \exp\left(-\int_{t_1}^t \sigma(r(u)) \cdot du\right) \quad (3)$$

通过追踪相机射线 $C(r)$ 通过待合成图像的每个像素来呈现新的视图。由于难以计算连续性函数，这个积分使用离散型数值方法计算。即使用了一种分层采样方法，其中射线被分成 N 个等间距距离，并从每个间隔中均匀抽取一个样本。则式 (2) 可近似为。

$$\hat{C}(r) = \sum_{i=1}^N \alpha_i T_i c_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \quad (4)$$

δ_i 是样本 i 到样本 $i+1$ 的距离。 (σ_i, c_i) 是沿给定射线的采样点 i 计算的密度和颜色，在采样点 i 处合成的透明度/不透明度为：

$$\alpha_i = 1 - \exp(\sigma_i \delta_i) \quad (5)$$

利用累积透过率，可以计算出射线的预期深度。

$$d(r) = \int_{t_1}^{t_2} T(t) \cdot \sigma(r(t)) \cdot t \cdot dt \quad (6)$$

这可以近似于式 (4) 近似式 (2) 和式 (3)。

$$\hat{D}(r) = \sum_{i=1}^N \alpha_i t_i T_i \quad (7)$$

一些深度正则化方法，通过使用预期深度将密度限制为场景表面，或强制深度平滑。

对于每个像素，使用用平方误差光度损失来优化 MLP 参数。对于整个图像，损失函数为 L_2 损失。

$$L = \sum_{r \in R} \|\hat{C}(r) - C_{gt}(r)\|_2^2 \quad (8)$$

其中 $C_{gt}(r)$ 是与 r 相关的训练图像像素的颜色， r 是与待合成图像相关的射线。

NeRF 模型采用位置编码，极大地改善了渲染视图中的精细细节重构。以下位置编码 γ 应用于 3D 坐标 x (归一化为 [-1,1]) 和观看方向单位向量 d 。

$$\gamma(\nu) = (\sin(2^0 \pi \nu), \cos(2^0 \pi \nu), \dots, \sin(2^{N-1} \pi \nu), \cos(2^{N-1} \pi \nu)) \quad (9)$$

其中 N 为编码维数参数，经过实验测试表明，对于 3d 坐标 x 设为 $N = 10$ ，对于方向 d 设为 $N = 4$ 时取得效果较好。

3 本文方法

3.1 Instant-ngp 方法概述

2022 年 1 月，Muller 等人发表了名为 InstantNeural Graphics Primitives (Instant-NGP) 的 NeRF 最新研究成果，极大地提高了 NeRF 模型训练和推理速度。作者提出了一种学习参数多分辨率哈希编码，其与 NeRF 模型 MLPs 同时训练。他们还采用了先进的射线推进技术，包括指数步进、空白跳过、样本压缩。这种新的位置编码和相关的 MLP 优化实现大大提高了训练和推理速度和场景重建精度。只需几秒钟的训练，该模型就取得了与之前 NeRF 模型中数小时训练相似的结果。

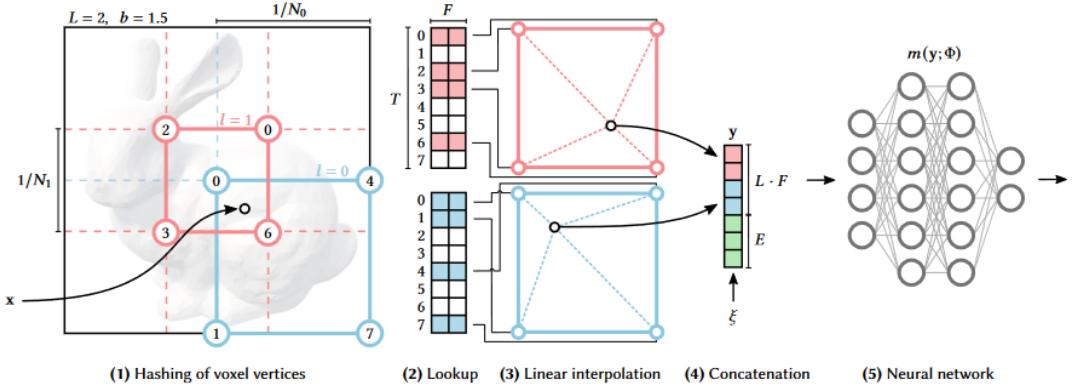


图 1: 二维多分辨率哈希编码的插图。(1) 对于给定的输入坐标 x , 在 L 分辨率水平上找到周围的体素, 并通过将整数坐标输入哈希函数得到分配的索引。(2) 对于所有得到的索引, 从哈希表 θ_l 中查找相应的 F 维特征向量。(3) 根据 x 在各自 1 层体素中的相对位置对它们进行线性插值。(4) 将每一级的结果和将 3D 坐标经过位置编码得到的值 $\xi \in R^E$ 进行拼接后输入到小型 MLP 网络 $\gamma \in R^{LF+E}$, (5) 最后得到 3D 点最终透明度和颜色。为了训练编码, 损耗梯度通过 MLP(5)、拼接(4)、线性插值(3)进行反向传播, 然后在查找的特征向量中积累。

Parameter	Symbol	Value
Number of levels	L	16
Max.entries per level(hash table size)	T	2^{14} to 2^{24}
Number of feature dimensions per entry	F	2
Coarsset resolution	N_{min}	16
Finest resolution	N_{max}	512 to 524288

表 1: 只有哈希表的大小为 T 取最大值时, 分辨率 N_{max} 需要针对任务进行调优。

3.1.1 位置编码

早期将机器学习模型的输入编码到高维空间的例子包括单热编码和核心技巧, 通过这种技巧, 数据的复杂排列可以线性可分。对于神经网络来说, 输入编码已被证明在循环架构的注意力组件中很有用, 随后是 transformer, 它们帮助神经网络识别它当前正在处理的位置。

在经典数据结构和神经方法之间, 有一种名为参数编码的方法, 其在实验中获得了最先进的结果。其想法是在辅助数据结构中安排额外的可训练参数(除了权重和偏差), 例如网格或树, 并根据输入向量 $x \in R^d$ 查找和插值(可选)这些参数。这种安排通过牺牲更大的内存占用来交换更小的计算成本。对于通过网络向后传播的每个梯度, 全连接 MLP 网络中的每个权值都必须更新, 对于可训练的输入编码参数(“特征向量”), 只有非常小的数量受到影响。例如, 对于特征向量的三线性插值三维网格^[16], 对于每个反向传播到编码的样本, 只需要更新 8 个这样的网格点。通过这种方式, 尽管参数编码的参数总数要比固定输入编码高得多, 但训练期间更新所需的参数数量和内存访问并不会显著增加。通过减小 MLP 层的大小, 这样的参数模型通常可以在不牺牲近似质量的情况下更快地训练收敛。

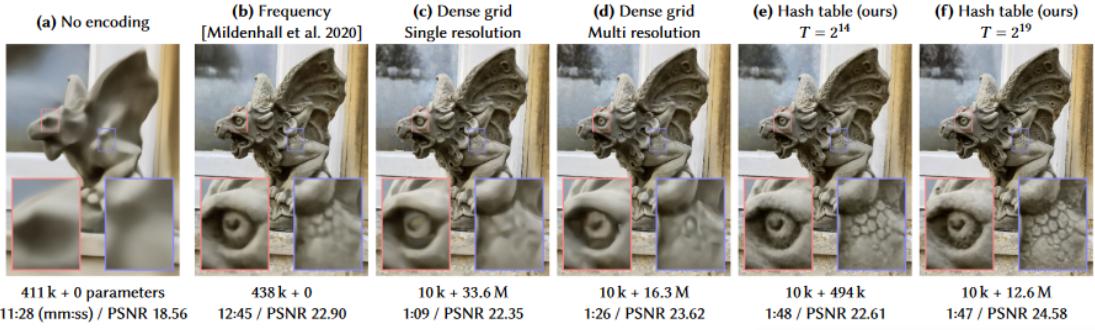


图 2: 用于存储可训练特征嵌入的不同编码和参数数据结构的重构质量。每个配置都使用 Instant-npg 实现并进行了 11,000 步的训练, 只改变了输入编码和 MLP 大小。可训练参数数 (MLP 权重 + 编码参数)、训练时间和重构精度 (PSNR) 在每张图像下方显示。Instant 的编码 (e) 具有与频率编码配置 (b) 相似的可训练参数总数, 由于参数更新的稀疏性和更小的 MLP, 训练速度超过 8 倍。增加参数数量 (f) 进一步提高了重建精度, 而不显著增加训练时间。

Instant-npg 的位置编码 $y = enc(x; \xi)$, 提高了近似质量和训练速度, 而不会产生显著的性能开销。在模型中, 编码参数 ξ 是可训练的。这些被安排到 L 层次, 每个层次包含维度为 F 的特征向量, 最多 T 大小。这些超参数的典型值如表 1 所示。图 1 说明了在多分辨率哈希编码中执行的步骤。每个层次 (图中红色和蓝色表示其中两个层次) 是独立的, 并在网格顶点上存储特征向量, 其分辨率被选择为最粗和最佳分辨率之间的几何级数 $[N_{min}, N_{max}]$:

$$N_l := \lfloor N_{min} \cdot b^l \rfloor \quad (10)$$

$$b := \exp\left(\frac{\ln N_{max} - \ln N_{min}}{L - 1}\right) \quad (11)$$

匹配训练数据中最好的细节时选择 N_{max} 。由于 L 数值众多, 增长因子通常很小, 一般是 $b \in [1.26, 2]$ 。

Hash 表的存取方式使用了空间哈希函数的形式

$$h(x) = \left(\oplus_{i=1}^d x_i \pi_i \right) \bmod T \quad (12)$$

其中, \oplus 表示按位异或运算, π_i 是唯一的大素数。这个公式按位异或运算每维线性同同排列的结果, 去关联维度对哈希值的影响。值得注意的是, 要实现独立性, d 维度中只有 d-1 维必须被排列, 因此选择 $\pi_1 := 1$ 以获得更好的缓存一致性, $\pi_2 = 2,654,435,761$, $\pi_3 = 805,459,861$ 。

3.1.2 射线推进

原始 NeRF 中将射线划分为等距离大小, 然后进行均匀采样。射线采样分为粗采样和细采样, 其中粗采样 64 个样本点, 细采样根据粗采样的结果进行优化采样位置, 在密度高的地方多采样, 采样 128 个样本点。但是在实际场景中, 大多数是空白的区域, 也就是大多射线的采样是没必要的。因此, 固定采样策略会浪费大量资源, 增加无用的耗时。

为了减少无用的采样操作和精准采样, Instant-npg 采用最先进的射线推进技术, 其中包括指数步进、空白跳过和样本压缩。

为了实现射线推进，论文采用根据场景大小变化的占用网格，通过 Ray Matching 查询射线与网格的交点。射线采样采用指数步进策略，对离相机近的场景多采样，离相机远的场景减少采样次数，并根据交点情况和网格占用情况进行空白跳过和样本压缩。

在实验中，由于占用率的不同，每条射线的样本数量是可变的，因此要在固定大小的批次中包含尽可能多的射线，而不是从固定的射线计数构建可变大小的批次。

Method	Batch size	= Samples per ray	\times	Rays per batch
Ours:Hash	256 Ki	3.1 to 25.7		10 Ki to 85 Ki
Ours:Freq	256 Ki	2.5 to 9		29 Ki to 105 Ki
mip-NeRF	1 Mi	128 coarse + 128 fine		4 Ki

表 2: Instant-ngp 完整方法 (our: Hash)、频率编码 NeRF 的实现 (our: Freq.) 和 mip-NeRF 的批大小、每批射线的数量和每射线的样本数量。

3.2 NeRFREN 方法概述

NeRF 将虚拟图像建模为真实的几何图形，这将导致深度估计不准确，并在违反多视图一致性的情况下产生模糊的渲染，因为反射对象可能只在某些视点下被看到。因此 NeRF 的视图依赖只能处理像高光这样的简单反射，而不能处理像来自玻璃和镜子的复杂反射。在严重的反射中包含一个稳定的虚拟图像，NeRF 倾向于将反射面后面的场景几何模型建模为半透明的 (像雾)，并将反射几何图形视为出现在某些虚拟深度。原因是 NeRF 并没有通过改变从不同方向观看时反射面点的颜色来解释这个点的视图依赖性，而是通过利用它后面的所有空间点沿着相机光线通过体渲染得到正确的颜色来解释它，导致几何图形雾霭和物理错误的深度。

NeRFREN 在 NeRF 基础上将场景分为透射和反射两部分，并使用独立的神经辐射场对这两个部分进行建模，适用于模拟带有反射的场景。但是这种分解是高度欠约束的，所以 NeRFREN 利用几何先验和精心设计的训练策略来获得合理的分解结果

NeRFREN 不是用单一的神经辐射场来表示整个场景，而是用单独的神经辐射场来模拟场景的传输和反射部分。为了合成新的视图，将对应辐亮度场渲染的透射像 I_t 和反射像 I_r 以相加的方式组合，其中反射像 I_r 用学习到的反射分数 β 进行加权：

$$I = I_t + \beta I_r \quad (13)$$

以无监督的方式将场景分解为透射和反射分量是高度欠约束的，直接训练的结果是 (1) 让一个组件解释整个场景，而让另一个组件空着;(2) 两个组件都对整个场景进行建模;(3) 在 (1) 和 (2) 之间的某个位置，其中一个组件的一部分混合到另一个组件中，导致不完整的透射/反射几何结构。

为了合理的分解场景，提出以下三点假设：

- 假设 1: 反射分数只与透射分量有关，因为它表示反射面的材料;
- 假设 2: 由于现实场景中大部分反射镜几乎是平面的，因此传输分量具有局部光滑的深度图;
- 假设 3: 反射组件只需要简单的几何图形来呈现正确的渲染图，因为大多数时候我们只能从非常有限的观看方向看到反射图像。

这些假设通常适用于在现实世界中常见的平面或几乎平面反射表面，为假设 1 设计了特定的网络

架构, 为假设 2 应用了深度平滑先验, 为假设 3 应用了新的双向深度一致性约束。至于更有挑战性的情况(如镜子), 在这种情况下, 歧义无法通过特定的设计解决, 为此利用了一个交互设置, 在这个设置中, 可以利用少量用户提供的反射掩模来获得正确的分解

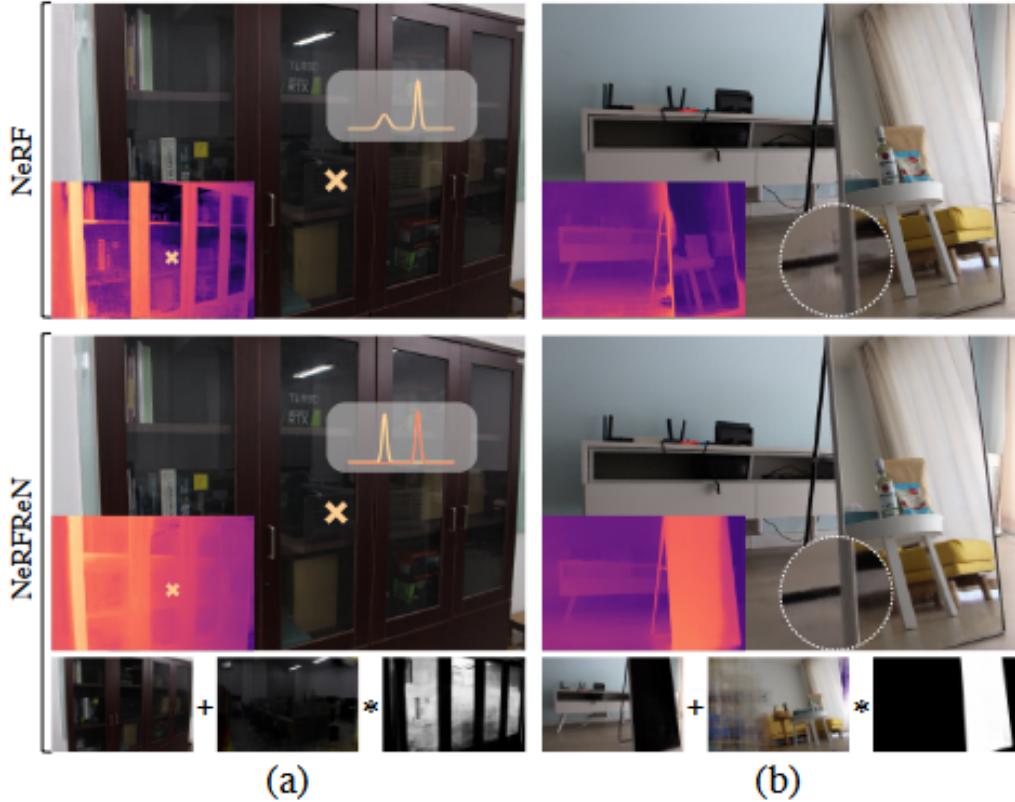


图 3: NeRF 将稳定的反射图像建模为真实的几何图形, 而不是反射表面上的点的视效。NeRF 在建模具有复杂反射的场景时可能遇到的两个问题:(1) 反射区域的深度估计不准确, 如 (a) 和 (b);(2) 违反多视图一致性时的不准确渲染, 如 (b) 中的放大区域。NeRFReN 通过使用单独的 nerf 对场景的传输和反射分量建模, 并在图像域中合成视图(下)来解决这些问题。

3.2.1 网络架构

NeRFREN 将场景分解为传输 NeRF 和反射 NeRF, 透射场的密度为 σ_t , 辐射度为 c_t , 反射场的密度为 σ_f , 辐射度为 c_f 。其外, 传输 NeRF 模型学习每个三维位置的反射分量值 α , 以测量物体在不同材料中的反射特性, 并通过体绘制累积像素对应的反射分量 β :

$$\beta(r; \sigma, \alpha) = \sum_k T_i(\sigma^t)(1 - \exp(-\sigma_i^t \delta_i)) \alpha_i \quad (14)$$

将反射色经 β 衰减后与透射色相加, 得到最终像素色

$$\hat{C} = \hat{C}(r; \sigma^t, c^t) + \beta(r; \sigma^t, \alpha) \hat{C}(r; \sigma^r, c^r) \quad (15)$$

反射分量值 α 与透射分支中的透射密度一起被预测, 因为它是反射表面的一种属性, 与反射分量无关。透射的颜色取决于观看方向, 因此低频视相关效果(如高光)由透射场建模, 稳定的虚拟图像由反射场建模。反射颜色和反射分量值不受观察方向的限制。由于假设 1 的有效性, 它们很好地近似了大多数场景, 并大大降低了网络的复杂性, 使不适当分解问题更容易学习

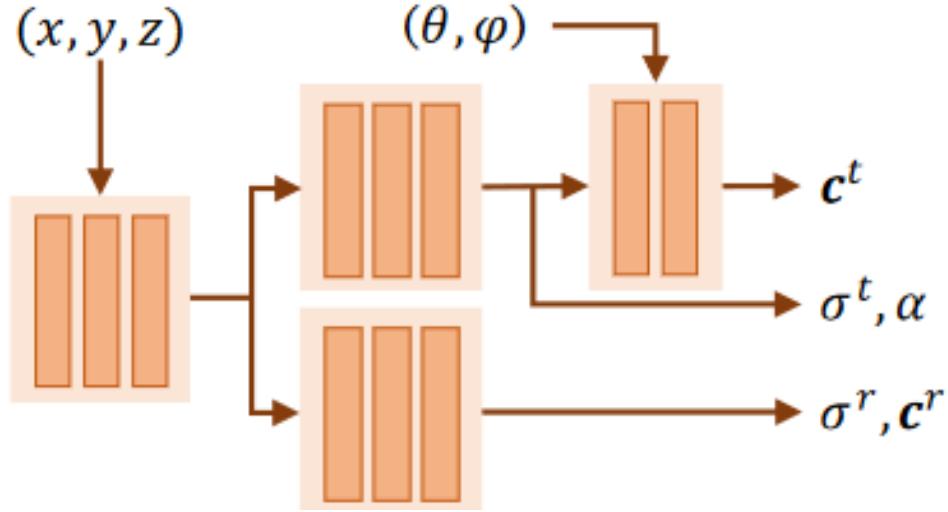


图 4: 传输和反射特性由网络的独立分支预测。反射分量值 α 是由透射分支预测的。

3.2.2 深度平滑

深度平衡利用了一个普遍的先验，即传输分量的深度映射应该是局部平滑的：

$$\ell_d = \sum_p \sum_{q \in N(p)} \omega(p, q) \|t^*(p) - t^*(q)\|_1, \quad \omega(p, q) = \exp(-\gamma \|C(p) - C(q)\|_1) \quad (16)$$

其中 $t^*(p)$ 是点的近似深度， p 表示图像中的每个像素， $N(p)$ 是其 8 个相邻像素的集合， C 是图像颜色， γ 是一个超参数， $\omega(p, q)$ 是一个衰减因子，用于基于颜色梯度重新加权约束，因为深度不连续常常伴随着突然的颜色变化，而通过 L_d 保持了边缘，只在小区域上工作，避免了在大多数情况下过度平滑。因为现实世界中的大多数反射面都是平面的，所以这一先验是合理的，在反射区域中显示平滑的深度变化。为了在训练过程中应用这种正则化，模型对补丁而不是像素进行采样。

3.2.3 双向深度一致性

深度平滑先验有利于传输部分的正确几何形状，这里假设有一个简单的几何形状来限制反射部分，其中每个射线只击中一个不透明的表面，如此只能从有限的视角看到被反射的物体，所以在某种程度上，它们类似于一个有纹理的固体表面。为此，定义一个新的后向深度 \overleftarrow{t}^* 沿射线 r 作为预期的终止点，看到来自 r 相反方向的体积：

$$\overleftarrow{t}^*(r; \sigma) = \sum_k \overleftarrow{\omega}_i t_i = \sum_k \overleftarrow{T}_i(\sigma) (1 - \exp(-\sigma_i \theta_i)) t_i \quad (17)$$

其中 $\overleftarrow{T}_i(\sigma) = \exp(-\sum_{j>i} \sigma_j \theta_j)$ ，双向深度一致性定义为：

$$L_{bdc} = \sum | \overrightarrow{t}^*(r; \sigma^r) - \overleftarrow{t}^*(r; \sigma^r) |_1 \quad (18)$$

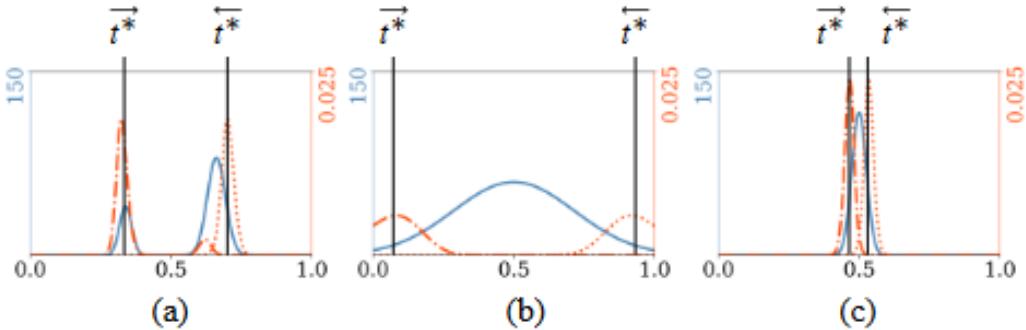


图 5: 双向深度一致性的插图。这里列出了神经辐射场中沿着射线的三种常见密度分布 (蓝色实线曲线)。对于每个图形，水平轴表示沿射线的点位置。点线和虚线橙色曲线是点权重，计算前后深度，并用黑线标出。只有在 (c) 有一个低的 BDC 值，这对应于预期的固体几何表面。

这种正则化对沿射线的密度分布提出了限制，迫使它是单峰的，并且具有小的方差。如图 5 所示，如果射线击中多个表面 (如 (a)) 或在“雾蒙蒙的”区域 (如 (b)) 没有清晰表面，则前后深度不一致。只有当密度分布为单模态且方差较小 (如 (c)) 时，两个深度值才会接近。直观地说，这要求反射分量具有“壳”样几何形状。

3.2.4 预热训练策略

用于优化 NeRFReN 的总损耗是光度损耗和提出的几何先验的加权组合：

$$L_{pm} = |\hat{C} - C|_2 \quad (19)$$

光度损失由预测颜色 \hat{C} 与实际颜色 C 的 L_2 损失得到。

$$L = L_{pm} + \lambda_d L_d + \lambda_{bdc} L_{bdc} \quad (20)$$

几何先验的大小会极大地影响分解结果，需要仔细调整权重因子，而这些权重因子很难跨场景共享，尤其在训练的早期，很难在光度损失和几何约束之间取得平衡，几何约束占主导地位，模型将停留在一个糟糕的局部极小值，只使用一个组件来解释整个场景，而留下另一个空白，如果光度损失占主导地位，几何正则化不足将导致次优分解结果。

为此，提出一种通用的训练策略，避免了对每个场景的超参数进行调优，有效地稳定了训练过程。对几何约束条件进行“热身”。 λ_d 和 λ_{bdc} 初始值较小，随着训练的进行， λ_d 和 λ_{bdc} 先增大后减小。

3.2.5 镜面交互设置

像镜子这样的无纹理反射器，在这种情况下，无监督分解通常会失败，可以利用额外的信息，如手动标记的反射分数映射。利用一种交互设置，其中用户提供少量训练图像的二进制掩码，1 和 0 分别表示反射和非反射区域。 L_1 损失用于鼓励预测反射分数映射 $\hat{\beta}$ 与用户提供的掩模之间的一致性。

$$L_\beta = \sum_p |\hat{\beta}(p) - \beta(p)|_1 \quad (21)$$

其中， $\hat{\beta} = \beta(r(p); \sigma^t, \alpha)$ 是反射分数值的估计值， $\beta(p)$ 是用户提供的二进制掩码在像素 p 处的值。

4 复现细节

NeRFREN 遵循原始 NeRF 纯 MLP 的网络结构设计，并采用粗网络和细网络结合的设计，因此存在网络训练时间长的缺点。而 Instant-npg 的多分辨率哈希编码、射线推进技术可以有效缩减 NeRF 训练时间，相对的，Instant-npg 和 NeRF 一样无法处理来自玻璃和镜子的复杂反射。为此在这里探究 Instant-npg 与 NeRFREN 结合效果。

多分辨率哈希编码方式为我们存储了大量角点特征向量，大大减少了 MLP 迭代层数，并且在反向传播的过程中减少了需要更新的参数量，与原始 NeRF 每一次反向传播都更新所有 MLP 权重信息先比，多分辨率哈希编码只需要更新小 MLP 权重信息和包含 3D 点所在位置的所有角点存储的特性向量即可。因此，在网络结构方面，选择了 Instant-npg 的多分辨率哈希编码结构。这里使用了 tinycudann 提供的 API，构建了一个层数为 16，每个角点存储 2 位向量，每层哈希数组大小为 2^{19} ，最小分辨率大小为 16，场景边框大小为 0.5。此外，经过多分辨率哈希编码后使用 ReLU 激活紧接上一个 1 层的小 MLP 输出 16 位对数值 h。由于反射分数值与投射分量无关，这里取 h 的第一位为透射密度值，第二位为反射分数值。3 位方向向量经过 4 阶球函数编码后得到 16 位位置编码值 d，将 16 位对数值 h 与 16 位位置编码值 d 拼接后经过 ReLU 函数后输入到 2 层使用 Sigmoid 激活的小 MLP，得到 3 位投射颜色值。而反射密度与反射颜色由一层小 MLP 使用 Sigmoid 激活后得到。

对于不同的数据场景要使用不同的场景边框进行射线步进采样。对于边框为 0.5 的小场景，使用边长为 1 的占用网格处理。将该网格边长划分为 128 份，总的快数大小为 2^21 。上文提到，场景中大多空间是空的，通过省去这些空间的计算可以大大减少无效采样点。为此，在网络训练前先进行占用网格的初始化，具体做法如下。首先将所有网格占用情况标记为 -1，即空白，通过使用数据集中的位置信息，将距离相机原点太近的网格和经过坐标变换后不处于图像坐标的网格去掉。之后的训练过程中，当迭代次数少于 256 次时进行全部网格占用情况更新，当迭代次数多于 256 次后，对剩余的占用网格进行抽样更新。每次更新中都从网络中获取 3D 位置的密度信息，根据密度阈值对网格占用情况进行更新。同时在每次训练迭代过程中，每条射线采样数量并不是固定不变的。通过利用最先进的指步进、空白跳过和样本压缩技术，每次射线都会根据占用网格进行 Raymatching，获取有效采样点。因此，在网络训练初期每条射线的采样点较多，随着网络的迭代，占用网格的逐步更新，射线的采样点逐渐减少，后期射线的采样点甚至缩减到 7 个左右。

实验的数据集为带有镜子的复杂反射，其中特意为镜子所在区域提供了掩码，为此针对该情况结合 NeRFREN 的反射场景处理技巧，主要计算以下损失。

由透射分量、反射分数值、反射分量计算得到预测颜色并与真实颜色进行 L_2 损失：

$$L_{pm} = |\hat{C} - C|_2 \quad (22)$$

$$\hat{C} = \hat{C}(r; \sigma^t, c^t) + \beta(r; \sigma^t, \alpha) \hat{C}(r; \sigma^r, c^r)$$

透射分量预测颜色与真实颜色进行 L_2 损失:

$$L_{pm} = |\hat{C}(r; \sigma^t, c^t) - C|_2 \quad (23)$$

分别对透射分量和反射分量计算深度平滑损失:

$$\ell t_d = \sum_p \sum_{q \in (p)} \omega(p, q) \|t_t^*(p) - t_t^*(q)\|_1 \quad \ell r_d = \sum_p \sum_{q \in (p)} \omega(p, q) \|t_r^*(p) - t_r^*(q)\|_1 \quad (24)$$

对反射分量计算双向深度一致性损失:

$$L_{bdc} = \sum | \vec{t}_r^*(r; \sigma^r) - \overleftarrow{t}_r^*(r; \sigma^r) |_1 \quad (25)$$

基于镜面大多为平面，所以相邻像素数值变化不大，对透射分量值计算平滑损失:

$$L_{beta} = \exp(-\gamma \|\beta(r; \sigma^t, \alpha)\|_1) \quad (26)$$

计算反射分数与掩码一致性损失:

$$L_\beta = \sum_p |\hat{\beta}(p) - \beta(p)|_1 \quad (27)$$

4.1 与已有开源代码对比

本次实验使用 python 开发，为了提升整体运行效率，减少无谓的时间浪费，部分射线推进技术引用了网络博主开源的 cuda 代码，射线推进技术部分需要大量烦杂的计算，使用 cuda 实现可以有效加速模型训练。

```

1 | class RayAABBIntersector(torch.autograd.Function):
2 |     @staticmethod
3 |     @custom_fwd(cast_inputs=torch.float32)
4 |     def forward(ctx, rays_o, rays_d, center, half_size, max_hits):
5 |         return vren.ray_aabb_intersect(rays_o, rays_d, center, half_size,
6 |                                         max_hits)
7 |
8 | class RayMarcher(torch.autograd.Function):
9 |     @staticmethod
10 |     @custom_fwd(cast_inputs=torch.float32)
11 |     def forward(ctx, rays_o, rays_d, hits_t,
12 |                 density_bitfield, cascades, scale, exp_step_factor,
13 |                 grid_size, max_samples):
14 |         noise = torch.rand_like(rays_o[:, 0])
15 |         rays_a, xyzs, dirs, deltas, ts, counter = \
16 |             vren.raymarching_train(
17 |                 rays_o, rays_d, hits_t,
18 |                 density_bitfield, cascades, scale,
19 |                 exp_step_factor, noise, grid_size, max_samples)
20 |         total_samples = counter[0] # total samples for all rays
21 |         xyzs = xyzs[:total_samples]
22 |         dirs = dirs[:total_samples]
23 |         deltas = deltas[:total_samples]
24 |         ts = ts[:total_samples]
25 |         ctx.save_for_backward(rays_a, ts)
26 |         return rays_a, xyzs, dirs, deltas, ts, total_samples
27 |
28 |     @staticmethod
29 |     @custom_bwd
30 |     def backward(ctx, dL_drays_a, dL_dxxyzs, dL_ddirs,
31 |                 dL_ddeltas, dL_dts, dL_dttotal_samples):
32 |         rays_a, ts = ctx.saved_tensors
33 |         segments = torch.cat([rays_a[:, 1], rays_a[-1:, 1] + rays_a[-1:, 2]])
34 |         dL_drays_o = segment_csr(dL_dxxyzs, segments)
35 |         dL_drays_d = \
36 |             segment_csr(dL_dxxyzs * rearrange(ts, 'n -> n 1') + dL_ddirs, segments)
37 |         return dL_drays_o, dL_drays_d, None, None, None, None, None, None, None
38 |
39 | class TruncExp(torch.autograd.Function):
40 |     @staticmethod
41 |     @custom_fwd(cast_inputs=torch.float32)
42 |     def forward(ctx, x):
43 |         ctx.save_for_backward(x)
44 |         return torch.exp(x)
45 |
46 |     @staticmethod
47 |     @custom_bwd
48 |     def backward(ctx, dL_dout):
49 |         x = ctx.saved_tensors[0]
|         return dL_dout * torch.exp(x.clamp(-15, 15))

```

本次实验重点在于实现 Instant-npg 与 NeRFREN 的结合，在尽量保持 Instant-npg 高效训练的特性的同时，优化反射场景渲染效果。本次实验在借鉴 Instant-npg 源码的基础上完成。主要完成了 rffr 数据处理、深度学习网络改进、损失函数改进和训练策略改进。实验中为 rffr 数据建立了一个专门的数据处理模块，完成图片、掩膜的读取、姿态的计算和射线的生成、获取等。由于 Instant-npg 采用了多分辨率哈希编码，使用空间存储信息而不是网络，因此大大减少了全连接层的层数，考虑到 NeRFREN 为渲染反射场景专门设计的网络结构，因此在实验中测试了不同网络结构。同样的，Instant-npg 的损失函数是 L2 损失，无法满足反射场景模型训练需求。为此结合 NeRFREN 为 Instant-npg 设置了 7 个损失函数。在完成上述工作后，发现模型能输出大概的深度图，但是渲染效果不理想，透射分量与反射分量相互交错，导致场景重现质量不佳，为此，在 NeRFREN 的训练策略的基础上寻找合适本模型的训练策略。实验中复现的 Instant-npg 在单张 3090 显卡上训练 Blender 的 Lego 模型需要 3 分钟左右，NeRFREN 则需要 2 天左右完成训练。实验完成的 Instant-npg 与 NeRFREN 的结合训练所需时间为 10 分钟左右，将 NeRFREN 训练时间提升到分钟级。

4.2 实验环境搭建

在本次实验中，该模型运行中 Ubuntu18.04, cuda11.3, Anaconda3-2022.10-Linux-x86_64, python3.8 环境上，并准备好 Synthetic_NeRF 数据并放在项目根目录/data 文件夹下后。同时需要在 python 环境上安装以下依赖包。

```
1 | torch-scatter
2 | tinycudann
3 | apex
4 | einops==0.4.1
5 | kornia==0.6.5
6 | pytorch-lightning==1.7.7
7 | matplotlib==3.5.2
8 | opencv-python==4.6.0.66
9 | lpips
10 | imageio
11 | imageio-ffmpeg
12 | jupyter
13 | scipy
14 | pymcubes
15 | trimesh
16 | dearpygui
```

最后在根目录下运行以下代码将 cuda 程序代码导入依赖。

```
1 | pip install models/csrc/
```

4.3 界面分析与使用说明

部署好运行环境后，在项目根目录运行以下命令即可启动项目，待项目运行完毕，在根目录下会生成 result 文件夹并存放实验结果。

```
1 | --root_dir ./data/Synthetic_NeRF/Lego --exp_name Lego
```

4.4 创新点

本实验主要利用了 Instant-npg 的多分辨率哈希编码技术和射线推进技术来优化可处理场景存在大量反射的数据集的 NeRFREN，将训练时间以小时为单位的 NeRFREN 提升到以分钟为单位。使用

了包括平滑损失、深度平滑损失和光度损失，对于放射严重场景加入了掩膜，并对掩码增加了平滑损失。

5 实验结果分析

在本次实验中使用了 pytorch lightning 进行代码复现，其中对资源、时间要求严格的部分如：射线步进技术部分和体渲染计算部分借鉴了 github 上的公开源码。在验证复现结果的过程了对 Blender 的 Lego 数据集和现实生活中使用红米 K30 手机随意拍摄的 35 张玩偶图片，其中随机选取 30 张用于训练，剩余 5 张用于测试（使用 colmap 估计相机位置，同时没有提供任何准确相机参数）。本次实验使用单张 3090 的 GPU，以 Lego 为代表的有界小场景上训练并渲染共花费了 3 分 46 秒，在个人数据集上训练并渲染共花费了 5 分 34 秒。从实验结果上看，实验成功将 NeRF 以小时为单位的训练时间提升到了以分钟为单位，虽然距离原 Instant-ngp 的 5 秒渲染速度有一定的差距，但是从结果上看这是可以接受的。实验数据及其结果如下所示：

Method	Dataset	avg PSNR	GPU	time
mine	Lego	38.40	3090	3m46s
instant-ngp	Lego	33.18	3090	5s

表 3: 不同数据集与原文效果对比

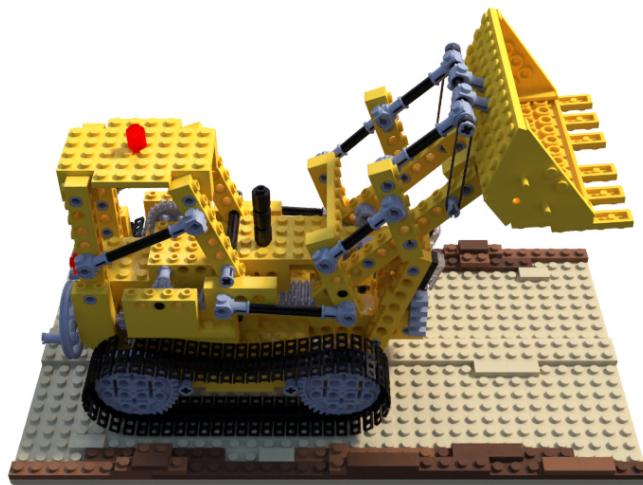


图 6: 复现代码 Lego 数据渲染结果



图 7: 复现代码个人拍摄数据渲染结果

6 总结与展望

在本次代码复现实验中，选择的对象是 Nvidia 于 2022 年发表的 Instant-ngp，该论文着重于 NeRF 训练时间优化，并成功将需要耗时 2 天训练的 NeRF 模型优化到 5s。其主要优化了三方面内容。第一，使用 cuda 语言编程，将代码加速做到极致；第二，使用多分辨率哈希编码，在保证保真度的同时大大减低了网络全连接层层数，并有效减少了参数更新。第三，使用最先进的射线推进技术，包括指数步进、空白跳过和样本压缩，这改变了 NeRF 固定采样策略，达到了精准采样，采样点随着训练过程越来越少，采样精准度也随着提升，因此调用网络次数减少，达到训练加速的目的。

在本次实验中选择了更为通用的 pytorch lightning 实现模型的总体架构，得到更清晰、更易学习的代码结构。但是单纯使用 pytorch lightning 实现会导致模型训练时间增加，减低实验效果，为此，射线推进技术部分和体渲染公式计算部分采用了 cuda 代码编写。经过实验证明，复现的代码能将 NeRF 训练时间提升到 3 分钟左右，虽然无法达到原始论文中的惊人效果，但是可以满足需求。

在项目改进方向上，主要着力于增加模型泛化性。计划提高 NeRF 对反射严重的场景的适用性，主要参考了《Is Attention All NeRF Needs?》(GNT) 中 Transformer 结构和《NeRFReN: Neural Radiance Fields With Reflections》中的平滑损失函数和深度平滑损失函数。在实验改进过程中发现，由于 Instant-ngp 的射线推进技术是不稳定采样，每条射线的采样数量不同，尤其在训练后期，射线间采样点数量差距较大，这种数据结构不适合使用 GNT 中所提出的 Transformer 结构。为此放弃采用 GNT 的模型结构。而在借鉴 NeRFReN 中的平滑损失和深度平滑损失的过程中，发现平滑损失和深度平滑损失对光度损失在训练的不同阶段有不同的影响，因此需要为训练过程制定严格的训练策略，这个过程是漫长且繁琐的，在短期内无法完全复现 NeRFReN 中惊艳的反射建模效果。

总的而言，本次复现实验并没有达到满意的效果，主要原因在于前期指定计划时没有深入了解技术细节，导致后期进行改进的时候才发现这条路行不通，最终导致复现成果不理想。为此在以后的研究中要加强细节的把控，同时要提高指定计划的能力，改进指定计划的方法，要充分考虑计划实施的可能性与计划的时间安排等。同时在代码复现过程中，要紧密联系论文，不能随意脱离原文发挥，要在完成基础任务后进行下一步的改进，而不是随心所欲，一边复现一边修改，因为在复现的过程中进

行改进可能会与后面的模块冲突，导致时间和精力的白白浪费。最后，在现阶段还没有足够的知识积累，可能在复现的过程中遇到意想不到的问题，要及时请教师兄师姐、老师或同学，要以最终目的为重，要善于从前辈身上学习。

在对 Instant-npg 代码复现的过程中，发现其多分辨率哈希编码和射线步进技术限制了场景的大小，并且在大场景中效果会下降，甚至会出现部分区域的缺失，噪音增多等情况，此外，Instant-npg 泛化性问题仍需借鉴更多更好的方案。未来可从这两部分对 Instant-npg 进行研究改进。

参考文献

- [1] MILDENHALL B, SRINIVASAN P, P, TANCIK M. Nerf: Representing scenes as neural radiance fields for view synthesis[J]. ACM Transactions on Graphics, 2021, [J]: 99-106.
- [2] STEPHAN J, MAREK K, MATTHEW J, et al. FastNeRF: High-Fidelity Neural Rendering at 200FPS [J]. arXiv, 2021, [J]: 2103.10380.
- [3] WADHWANI K, KOJIMA T. SqueezeNeRF: Further factorized FastNeRF for memory-efficient inference[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 2717-2725.
- [4] YU A, FRIDOVICH-KEIL S, TANCIK M. Plenoxels: Radiance fields without neural networks[J]. arXiv preprint, 2021, [J]: 18409-18418.
- [5] MÜLLER T, EVANS A, SCHIED C, et al. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding[J]. arXiv preprint, 2022, [J]: 05989.
- [6] NIESSNER M, ZOLLMÖFER M, IZADI S. Real-time 3D reconstruction at scale using voxel hashing [J]. ACM Transactions on Graphics, 2013, [J]: 1-11.
- [7] BARRON T, J, MILDENHALL B, TANCIK M. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields[J]. arXiv preprint, 2021, [C]: 5855-5864.
- [8] BARRON T, J, MILDENHALL B, VERBIN D. Mip-nerf 360: Unbounded anti-aliased neural radiance fields[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 5470-5479.
- [9] REMATAS K, LIU A, SRINIVASAN P, P. Urban radiance fields[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 12932-12942.
- [10] TANCIK M, CASSER V, YAN X. Block-nerf: Scalable large scene neural view synthesis[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 8248-8258.
- [11] XIANGLI Y, XU L, PAN X. Citynerf: Building nerf at city scale[J]. arXiv preprint, 2021, [J]: 2112.05504.
- [12] CHEN X, ZHANG Q, LI X. Hallucinated neural radiance fields in the wild[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 12943-12952.
- [13] HUANG X, ZHANG Q, FENG Y. HDR-NeRF: High Dynamic Range Neural Radiance Fields[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 18398-18408.
- [14] GUO C, Y, KANG D, BAO L. Neural radiance fields with reflections[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 18409-18418.
- [15] XU Q, XU Z, PHILIP J. Point-nerf: Point-based neural radiance fields[J]. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, [C]: 5438-5448.
- [16] SAEED H, SHUHONG C, MATTHIAS Z. Neural radiosity. ACM Transactions on Graphics[J]. ACM Transactions on Graphics, 2021, [J]: 1-11.