文章编号:1007-5321(2021)06-0109-07 **DOI**:10.13190/j.jbupt.2021-095

一种联合采样的神经网络光场

刘绍华[1], 李明豪[1,2], 李兆歆[2], 毛天露[2], 刘 京[3]
(1. 北京邮电大学 信息物理融合系统研究实验室, 北京 100876; 2. 中国科学院 计算技术研究所, 北京 100190;
3. 河北师范大学 软件学院, 石家庄 050024)

摘要: 相比传统的光场绘制技术,神经网络光场(NeRF)方法可使用神经网络拟合场景的光线采样,将隐式编码输入图片的光场,合成新视图. 针对 NeRF 方法训练时间长,绘制视图慢的问题,提出了一种基于联合采样的 NeRF 方法,通过使粗糙网络和细腻网络共享均匀采样结果的方法,减少了不必要的光线采样,从而加快了网络训练和视图合成的速度. 实验结果表明,在取得近似视图合成质量的情况下,与 NeRF 方法相比,所提方法的训练时间减少了 20%,视图合成的效率提高了 25%.

关 键 词: 神经网络光场; 光线采样; 联合采样; 视图合成
中图分类号: TN911.22 **文献标志码**: A

Neural Radiance Field by Joint Sampling

LIU Shao-hua[1], LI Ming-hao[1,2], LI Zhao-xin[2], MAO Tian-lu[2], LIU Jing[3]
(1. Cyber-Physical Systems Laboratory, Beijing University of Posts and Telecommunications, Beijing 100876, China;
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;
3. School of Software, Hebei Normal University, Shijiazhuang 050024, China)

Abstract: Compared with the traditional light field, the neural reflectance field (NeRF) method uses the neural network to fit the light sampling of scenes, which implicitly encodes the light field from input images to render novel view. However, NeRF method requires long training time and has slow rendering speed. To solve this problem, a joint sampling-based NeRF is proposed, which can make the coarse network and fine network share uniform sampling results, thereby accelerating the network training and view synthesis by reducing unnecessary light sampling. The experiments demonstrate that, in the case of the similar view synthesis quality, compared with the baseline method, the proposed method can reduce the training time by 20% and improve the view synthesis efficiency by 25%.

Key words: neural reflectance field; light sampling; joint sampling; view synthesis

光场(LF, light field)绘制[1]是一种新的便捷、可动态呈现真实感场景画面的技术,不需要专业人员设计复杂的几何模型,也不需要精细的纹理贴图和光照模拟,只要在多个视点位置上拍摄一定数量的照片,就可以在未拍摄的新视点利用原始图片直接合成逼真的场景视图. 光场绘制的原理是尽可能地采集场景空间中的光线信息,然后使用这些光线信息再还原出某一位置和角度观察到的场景视图. 由于传统光场绘制技术[2]使用启发式的采样方式,往往需要捕获密集和规则的照片,难以广泛应用于

收稿日期: 2021-05-23
基金项目: 国家自然科学基金项目(91938301, 62172392)
作者简介: 刘绍华(1976—), 男, 副教授, 博士生导师.
通信作者: 李兆歆(1983—), 男, 助理研究员, E-mail: cszli@hotmail.com.

实验室外的场景.

近年来,一些研究将深度学习技术用于光场绘制. 如神经体绘制[3]使用神经网络学习场景的一种半透明体表示,从而利用多视点图像呈现动态场景. 局部光场融合[4]通过学习输入图片的多平面图像进行绘制,使手持拍摄的稀疏图片也能生成较好的新视图. 神经稀疏体素场[5]使用一种可微的光线行进策略学习场景的稀疏体素表示,将体素与光场相结合,可以自由编辑组合场景内容的同时生成逼真的场景画面,但受限于有界场景. 神经网络光场(NeRF,neural reflectance field)[6]方法使用多层感知机(MLP,multi-layer perceptron)网络隐式地编码输入图片中的场景光线,然后通过分层采样近似积分可以绘制出高质量的目标视图. NeRF 方法适用于从虚拟合成到真实世界手持拍摄的各种数据集,并都取得了高质量的视图合成结果. 但由于其需要隐式地拟合场景中所有光线的采样情况,目前还普遍存在训练时间长、运算速度慢的问题.

Lindell 等[7]通过修改积分公式,降低合成像素颜色的计算复杂度,以加快 NeRF 方法在测试时的效率. Reiser 等[8]使用上千个小型 MLP 网络分别表示场景的不同区域,提高视图合成速度,但其需要先训练一个收敛的 NeRF 方法,然后得到这些小型 MLP 网络的参数. 上述方法的共同点是只能改善训练完成后 NeRF 方法在测试阶段的效率,而无法减少训练模型所需要的时间. 为此,提出一种联合采样的神经网络光场(NeRF-JS,neural radiance field by joint sampling)方法,通过使粗糙网络和细腻网络共享均匀采样的结果,协同合成新视图,以减少训练和测试整个过程的计算量.

目前的 NeRF 方法使用粗糙和细腻双网络级联的采样方案来拟合空间光场. 其中,通过粗糙网络执行均匀采样,可以得到一束光线上有效采样位置的分布,从而拟合这束光线上有效采样位置的概率密度函数. 基于这个概率密度函数,细腻网络在概率大的位置执行更稠密的非均匀采样,从而拟合更精细的空间光场. 在 NeRF 方法中,只有细腻网络的采样结果可以被用来近似积分合成新视图. 因此,为了保证最终的视图合成质量,其中的细腻网络也需要执行一次均匀采样. 这种采样方法中,执行了 2 次均匀采样.

在 NeRF-JS 方法中,均匀采样只需由粗糙网络采样一次,最终的视图合成中除了使用细腻网络输出的非均匀采样结果,同时还使用已获得的粗糙网络采样结果,均匀采样只需由粗糙网络执行一次,相比原始的 NeRF 方法在理论上节省了细腻网络均匀采样的计算量. NeRF-JS 与原始 NeRF 方法两者的技术实现方式如表 1 所示.

表 1 NeRF-JS 与原始 NeRF 方法的对比

技术实现方式	原始 NeRF 方法	NeRF-JS
网络结构类型	MLP	MLP
粗糙-细腻结构	是	是
网络间采样方式	分离	联合
均匀采样次数	2	1
粗糙网络参与颜色生成	否	是
细腻网络参与颜色生成	是	是

实验结果证明,在相同的粗糙和细腻采样精细度下,NeRF-JS 方法在保证与原 NeRF 方法近似视图质量的同时,节省了 20% 的训练时间,提高了 25% 的视图合成效率.

1 神经网络光场

光场绘制的理论基础是一个五维全光函数[9]. 函数的输入是三维位置 $\boldsymbol{x}=(x,y,z)$ 和基于球坐标系的方向向量 $\boldsymbol{d}=(\theta,\phi)$. 函数的输出是光线沿着方向 $\boldsymbol{d}$ 到达三维位置 $\boldsymbol{x}$ 所呈现的颜色 $\boldsymbol{c}$ 和体密度 σ. NeRF 方法通过 MLP 网络隐式地拟合这个场景的五维全光函数:$F_{\Theta}(\boldsymbol{x},\boldsymbol{d})$. 神经网络的训练过程就是不断地调整网络模型的权重参数 $\boldsymbol{\Theta}$,使其最终可以在给定输入后,输出与实际光线一致的颜色 $\boldsymbol{c}$ 和体密度 σ.

在场景中拍摄照片,实质上是根据一定规则采集摄像机观察到的所有光线,并使用感光元件记录最终到达感光平面颜色的过程. 根据经典的体渲染理论,可以用空间采样,然后积分的方法得到一束光线在摄像机上呈现的颜色值 $C(\boldsymbol{r})$[6]. 一束光线可以用 $\boldsymbol{r}(t)=\boldsymbol{o}+t\boldsymbol{d}$ 表示,其中 $\boldsymbol{o}$ 为光线起点,t 为沿着光线方向与起点的采样距离. 对于从近平面 t_{n} 到远平面 t_{f} 的一束光线 $\boldsymbol{r}$,其在投影到近平面上呈现的颜色值为

$$C(\boldsymbol{r})=\int_{t_{\mathrm{n}}}^{t_{\mathrm{f}}}T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t))\,\mathrm{d}t\text{[6]}\quad(1\mathrm{a})$$

$$T(t)=\exp\left(-\int_{t_{\mathrm{n}}}^{t}\sigma(\boldsymbol{r}(s))\,\mathrm{d}s\right)\text{[6]}\quad(1\mathrm{b})$$

其中:$T(t)$ 为从 t_{n} 到 t 累积的透光度,即光线从 t_{n}

到 t 位置过程中没有被遮挡或者阻断的概率. 而光线采样位置的体密度 σ 可以看作为光线在某一无穷小位置处终止的微分概率. 将摄像机的感光元件平面设置为近平面,无穷远处设为远平面,可计算摄像机画面中每个像素对应的光线积分颜色,继而合成整张视图. 式(1a)对应的是一个连续的无限密度采样,使用计算机采样时,还需要用离散采样求和的方式来拟合连续采样的积分,即将一束光线从 t_n 到 t_f 的距离划分为 N 个均匀分布的区间,每个区间作为一个采样位置[6]:

$$[t_n,t_f]\rightarrow \sum_{i=1}^{N}\left[t_n+\frac{i-1}{N}(t_f-t_n),t_n+\frac{i}{N}(t_f-t_n)\right] \tag{2}$$

拟合值和真实值的相似度与采样数量成正比,即在其他条件一致的情况下,离散采样的数量越多,拟合的光线颜色越准确,合成视图的质量越好. 基于离散的区间划分,光线连续积分可以转化为[6]

$$\hat{C}(\boldsymbol{r})=\sum_{i=1}^{N}T_i[1-\exp(-\sigma_i\delta_i)]c_i \tag{3a}$$

$$T_i=\exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \tag{3b}$$

其中:$\delta_i=t_{i+1}-t_i$,为第 i 个采样区间的长度,c_i 和 δ_i 为神经网络在第 i 个采样位置输出的颜色和体密度. 由(c_i,δ_i)计算 $\hat{C}(\boldsymbol{r})$的过程类似于传统使用 Alpha 透明度的混合渲染,透明度 $a_i=1-\exp(-\sigma_i\delta_i)$. 摄像机拍摄画面中的每一个像素对应一束从摄像机光心发出的光线,通过将神经网络采样积分得到的像素颜色作为预测值,输入图片像素的真实颜色作为真实值,可以计算损失函数,从而迭代训练神经网络.

然而,基于以上光线采样原理构建的简单神经网络并不能合成高质量的视图. NeRF 方法采取了 2 种措施来提高视图合成质量.

1) 对输入进行扩张维度编码. NeRF 方法直接将位置和方向组成的五维向量作为输入的网络模型进行实验,发现其在渲染颜色和几何变化较多的高频内容方面效果较差. Rahaman 等[10]证明了深度神经网络更倾向于学习低频函数,而在进入网络之前,预先使用高频函数将输入映射到更高的维度可以更好地拟合高频变化的数据. 因此,NeRF 方法可对输入的位置和方向预先进行扩张维度的编码,然后再输入神经网络. 具体来说,对于一维值 p,其编码方式为[6]

$$\gamma(p)=(\sin(2^0\pi p),\cdots,\sin(2^{L-1}\pi p),\cos(2^{L-1}\pi p)) \tag{4}$$

其中:L 为扩张的维度,可影响模拟的细腻程度. 编码函数 $\gamma(p)$将 p 通过傅里叶变换扩展为一个 $2L$ 维的向量.

2) 粗糙细腻双网络的级联方案. 先使用一个粗糙网络进行均匀采样;然后使用一个细腻网络进行非均匀采样. 具体来说,对于一组均匀的采样位置(数量用 N_c 表示),可以将粗糙网络采样结果合成的光线颜色看作所有采样位置采样颜色的加权和[6]:

$$\hat{C}_c(\boldsymbol{r})=\sum_{i=1}^{N_c}w_ic_i \tag{5a}$$

$$w_i=T_i[1-\exp(-\sigma_i\delta_i)] \tag{5b}$$

每个采样位置的权重大小间接反映了其附近区域存在有效内容的可能性. 因此,可以将每个粗糙采样的归一化权重看作沿着光线存在有效点的概率密度函数. 通过对这个概率密度函数进行反采样,可以得到一组在高权重位置附近密集,在低权重附近稀疏的非均匀采样位置(数量用 N_f 表示). 然后,将之前的均匀采样位置和非均匀采样位置组合起来,利用细腻网络在这些位置再进行采样,其采样结果被用来合成最终视图像素对应的光线颜色[6]:

$$\hat{C}_f(\boldsymbol{r})=\sum_{i=1}^{N_c+N_f}T_i[1-\exp(-\sigma_i\delta_i)]c_i \tag{6}$$

需要注意的是,NeRF 方法中粗糙网络和细腻网络是分别独立的 2 个模型,网络的权重参数并不共享,最后用来合成目标视图的光线颜色只来自细腻网络.

2　联合采样优化的 NeRF

相比其他方法[3,4,11],NeRF 方法已经被证明在多种数据集上都取得了很好的视图合成效果. 但是,由于 NeRF 这种采样方法在每束光线上的有效采样数量决定了最终的视图合成质量,要获得好的视图质量,直接提高采样数量是有效的办法,但相应的计算代价也非常高昂. 原始的 NeRF 方法采用了一种级联的双网络架构,尽量在高可能性的位置采样,以提高视图质量. 其中,粗糙网络主要用来“探查”空间位置的权重分布,细腻网络则作为真正的颜色输出网络,在权重高的地方进行更密集的采样.

笔者提出一种联合采样的方案,将粗糙网络的均匀采样结果与细腻网络的非均匀采样结果联合起来,生成最终的像素颜色,而不是全部由细腻网络采样. 联合采样方案如图 1 所示.

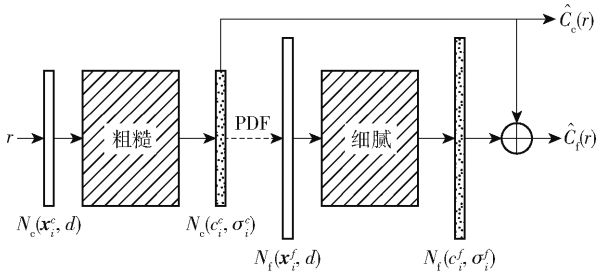

图1 联合采样方案示意图

具体来说,粗糙网络的均匀采样位置保持不变,其输出的均匀采样结果为

$$(c_i^c, \sigma_i^c) = F_\Theta^c(\boldsymbol{o} + i\Delta\boldsymbol{t}, \boldsymbol{d}), i = 1, 2, \cdots, N_c \quad (7)$$

其中:$\boldsymbol{d}$ 为所有采样位置的方向,$\Delta\boldsymbol{t}$ 为均匀采样的采样间隔,F_Θ^c 为粗糙网络的采样函数. 使用粗糙网络输出的体密度 σ_i^c,可以通过式(5b)计算每个位置的归一化权重:

$$\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_c} w_j} \quad (8)$$

权重分布可以作为概率密度函数执行反向采样,得到 N_f 个非均匀采样位置输入细腻神经网络中,得到细腻采样结果,有

$$(c_i^f, \sigma_i^f) = F_\Theta^f(\boldsymbol{x}_j^f, \boldsymbol{d}), j = 1, 2, \cdots, N_f \quad (9)$$

其中:F_Θ^f 代表细腻网络对应的采样函数,$\boldsymbol{x}_j^f$ 为非均匀采样位置. 对于光线 $\boldsymbol{r}$,粗糙网络采样结果的颜色计算方式如下:

$$\widetilde{C}_c(\boldsymbol{r}) = \sum_{i=1}^{N_c} T_i [1 - \exp(-\sigma_i^c \delta_i)] c_i^c \quad (10)$$

细腻网络输出的光线颜色由其自身输出的非均匀采样结果与粗糙网络的均匀采样结果混合而成. 但由于光线采样结果的累计需要由近至远进行,所以可使用 NeRF-JS 方法将非均匀和均匀的采样位置合在一起做一次重排序:

$$\{\boldsymbol{x}_{k=1,2,\cdots,N_c+N_f}\} = \mathrm{sort}(\{\boldsymbol{x}_{i=1,2,\cdots,N_c}^c\}, \{\boldsymbol{x}_{i=1,2,\cdots,N_f}^f\}) \quad (11)$$

其中 sort 为排序函数. 同时,将每一个 $\boldsymbol{x}_k$ 与各自的采样结果对应起来,σ_k 为 $\boldsymbol{x}_k$ 位置的体密度,c_k 为 $\boldsymbol{x}_k$ 位置的采样颜色. 然后联合2种采样结果计算出最终视图像素对应的光线颜色,有

$$\widetilde{C}_f(\boldsymbol{r}) = \sum_{k=1}^{N_c+N_f} T_k \{1 - \exp[-\sigma_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})]\} c_k \quad (12a)$$

$$T_k = \exp\left[-\sum_{k=1}^{k-1} \sigma_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})\right] \quad (12b)$$

其中:$\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ 为第 k 个采样位置与前一个采样位置的距离,$\boldsymbol{x}_0$ 的值为0.

最后,在每个像素上计算粗糙和细腻2种颜色结果的均方误差作为损失函数来同步训练2个网络,有

$$Q = \sum_{\boldsymbol{r}\in\mathbf{R}} (\| \widetilde{C}_c(\boldsymbol{r}) - C_{gt}(\boldsymbol{r}) \|_2^2 + \| \widetilde{C}_f(\boldsymbol{r}) - C_{gt}(\boldsymbol{r}) \|_2^2) \quad (13)$$

其中 $C_{gt}(\boldsymbol{r})$ 为光线对应像素颜色的真实值.

3 实验结果

3.1 数据集

在 Realistic Synthetic 360°数据集(合成场景)和 Real Forward-Facing 数据集(真实世界)上进行了实验,以验证所提联合采样优化算法的有效性.

如图2(a)所示,Realistic Synthetic 360°是为评估 NeRF 方法有效性提供的合成数据集,包括 Chair, Drums, Ficus, Hotdog, Lego, Materials, Mic 和 Ship 等8个由 Blender 软件精细建模的复杂非朗伯场景. 每个场景的图片由 Blender 的环形路径追踪插件采集,其中的6个场景是从上半球方向环绕拍摄的,另外2个场景是从整个球形的其他方向采集. 选取每个场景的100张图片用神经网络算法进行训练,其中的13张图片用于测试. 该数据集所有图片的分辨率为 800×800.

如图2所示,Real Forward-Facing 数据集包括

(a) Realistic Synthetic 360°数据集

(b) Real Forward-Facing数据集

图2 实验采用的数据集[4-6]

Room[6], Fern[4], Leaves[6], Fortress[4], Orchids[4], Flower[4], Trex[4] 和 Horns[6] 等 8 个真实世界的场景. 与 Realistic Synthetic 360°数据集不同,该数据集的图片是使用智能手机面对着场景上下左右平移拍摄的,即采用 Forward-Facing 视角拍摄. 由于是手持拍摄的真实场景,每张图片对应的摄像机参数由 colmap[12-13] 估计得到. 每个场景的图片数量为 20 ~ 62 不等,其中 1/8 的图片可作为测试集,其他图片作为训练集. 该数据集所有图片的分辨率为 1 008 × 756.

3.2　实验环境及参数设置

NeRF-JS 方法由 Python 语言基于 PyTorch 框架实现. 算法效率低的原因主要在于计算开销集中在神经网络计算光线采样结果,由 NVIDIA Tesla T4 GPU 进行运算.

算法涉及的参数分为以下 2 类.

1) 光线采样相关参数. 参数包括:粗糙网络进行均匀采样的位置数 $N_c = 64$;细腻网络的非均匀采样位置数 $N_f = 128$;根据式(4)对每一维位置向量进行扩张维度为 10 的编码;对每一维方向向量进行扩张维度为 4 的编码.

2) 神经网络训练相关参数. 训练时每次迭代的并行光线为 1 024 束,网络单次并行采样的数量为 16 384,以上 2 个参数可根据显卡显存的大小进行调节. 使用 Adam 优化器[14] 学习和更新权重参数,其超参数 $\beta_1 = 0.9, \beta_2 = 0.9, \varepsilon = 1 \times 10^{-7}$. 初始的训练学习率设为 5×10^{-4},并以 5×10^{-5} 指数递减,当前迭代的学习率为

$$\alpha_{cur} = 5 \times 10^{-4} \times 0.1^{\frac{q}{g}} \tag{14}$$

其中:q 为当前的迭代次数,g 为总的迭代次数,每个场景迭代训练 50 万次. 以上参数数值均与原始 NeRF 方法对应的参数值保持一致.

3.3　评价指标

在神经网络光场训练完成后,通过合成未参与训练的测试集图片来评估算法合成的视图质量. 在评价指标方面,选择使用峰值信噪比(PSNR, peak signal to noise ratio)和结构相似度(SSIM, structural similarity)两项得分来评估神经网络光场算法合成的目标视图和真实原图的差异.

另外,光线采样优化的主要目的在于减少计算量,提高速度. 因此实验中还用每张图片合成所需的时间和网络训练时间来衡量算法的效率.

3.4　实验结果对比与分析

下面将 NeRF-JS 方法与原始的 NeRF 方法进行对比. 首先,对 2 种方法的效率进行理论分析. 假设单个 MLP 网络执行单个采样位置的计算量为 τ,在每束光线均匀采样数量 $N_c = 64$,非均匀采样数量 $N_f = 128$ 的情况下,由于原始 NeRF 方法的细腻网络仍需再执行一遍均匀采样,其采样单束光线的总计算量为 256τ. 由于 NeRF-JS 方法的细腻网络不再执行 $N_c = 64$ 的均匀采样,其采样单束光线的总计算量为 192τ. 也就是说,NeRF-JS 方法在神经网络部分相比原始的 NeRF 方法在理论上可以节省 25% 的计算量.

在 Realistic Synthetic 360°数据集上用 2 种方法合成视图的质量及效率如表 2 所示,其中加粗的数字为最佳得分. 可以看出,相比原始 NeRF 方法,NeRF-JS 方法在减少 25% 计算时间的同时,保证了与原始方法几乎一致的视图合成质量.

表 2　在 Realistic Synthetic 360°数据集上合成视图的效果

场景	NeRF[6]			NeRF-JS		
	PSNR	SSIM	t/s	PSNR	SSIM	t/s
Chair	34.56	0.979	20.61	34.44	0.979	15.15
Drums	26.83	0.935	20.23	26.78	0.934	14.96
Ficus	30.15	0.973	20.82	30.18	0.973	15.24
Hotdog	36.70	0.977	21.07	36.77	0.979	15.33
Lego	33.07	0.972	21.69	33.11	0.971	15.61
Materials	30.36	0.965	21.05	30.52	0.967	15.51
Mic	33.83	0.980	21.10	33.90	0.980	15.53
Ship	31.08	0.879	20.92	30.93	0.877	15.40
平均值	32.07	**0.958**	20.94	**32.08**	**0.958**	**15.34**

在 Real Forward-Facing 数据集上用 2 种方法合成视图的质量及效率如表 3 所示,其中加粗的数字为最佳得分. 可以看出,与表 2 中的结果类似,NeRF-JS 方法在大幅度减少计算时间的同时,保证了与原始方法几乎一致的视图合成质量,在速度和质量两方面取得了均衡.

另外,由于 NeRF 方法本质上是使用权重参数对单一场景进行隐式编码,实际应用时需要在每个场景上重新训练模型. 因此,节省网络模型的训练时间也十分必要. 表 4 所示为在 2 个数据集上平均每个场景的训练时间. 可以看出,NeRF-JS 方法相较于 NeRF 方法减少了 20% 左右的训练时间. 之所以是 20% 而不是 25%,可能是因为在 ReRF 方法训

练过程中,光线采样的计算量只是一部分,还有数据加载等其他操作也需要耗费一定的时间.

表3 在 Real Forward-Facing 数据集上的效果

场景	NeRF			NeRF-JS		
	PSNR	SSIM	t/s	PSNR	SSIM	t/s
Fern	27.00	0.862	25.33	27.01	0.863	19.00
Flower	28.62	0.890	28.12	28.70	0.891	21.05
Fortress	33.04	0.932	28.75	33.01	0.932	21.57
Horns	29.50	0.913	26.80	29.27	0.905	20.01
Leaves	22.56	0.812	25.22	22.56	0.812	19.98
Orchids	21.24	0.740	25.17	21.34	0.742	19.15
Room	33.71	0.963	25.40	33.76	0.963	19.09
Trex	28.47	0.932	26.02	28.51	0.932	21.05
平均值	**28.02**	**0.881**	26.35	**28.02**	0.880	**20.11**

表4 平均每个场景的训练时间 h

数据集	NeRF	NeRF-JS
Realistic Synthetic 360°	52.37	**40.17**
Real Forward-Facing	56.90	**43.59**

3.5 视图结果对比

除了基准方法外,NeRF-JS 方法还通过减少原始 NeRF 方法中细腻网络的均匀采样数量进行消融. 其中,粗糙网络的均匀采样数量仍为64,并用于反采样,得到128个非均匀采样位置,只是细腻网络再次执行均匀采样的采样位置数量由64变更为32. 这种方案命名为"NeRF-lite",其计算量为224τ,介于 NeRF 方法与 NeRF-JS 方法之间,即理论上可以节省12.5%的神经网络计算开销.

图3所示为在2种数据集上的视图合成结果,其中,第1列为数据集提供的真实值,第2列为NeRF方法的合成结果,第3列为NERF-JS方法的合成结果,第4列为NeRF-lite方法的合成结果. 图3中第1行的结果来自 Real Forward-Facing 真实世界数据集的 Trex 场景,第2和第3行显示了图片的放大细节. 可以看出,在红色放大部分,NeRF-JS 方法得到了最为清晰的细节(黑色顶灯及护栏). 在绿色放大部分,NeRF-JS 重建出了一部分栏杆和蓝色挂绳,用其他2种方法则都比较模糊. 第4行的结果来自 Realistic Synthetic 360°数据集的 Drums 场景. 从最后一行的放大细节可以看出,NeRF-lite 的结果在鼓面交界处出现了明显的腐蚀,而 NeRF-JS 方法与原始 NeRF 方法的结果则相差不多.

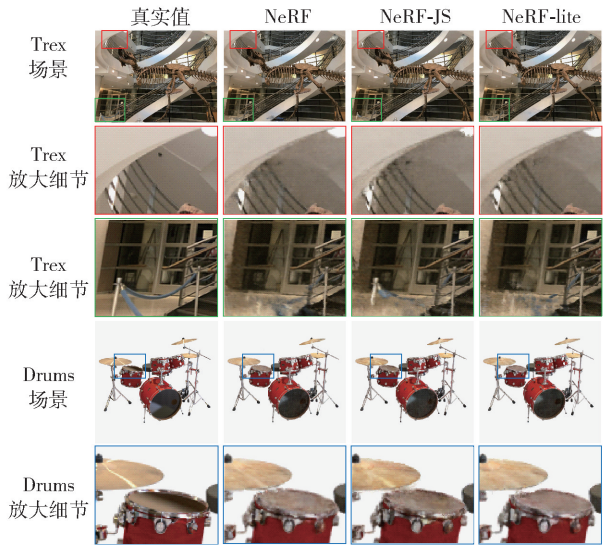

图3 视图合成结果对比

4 结束语

提出了一种基于联合采样优化的神经网络光场方法,通过在级联网络间共享均匀采样结果来减少计算量. 实验结果证明,相比原始 NeRF 方法,NeRF-JS 方法可以在保证同等视图质量的同时,减少20%左右的训练时间,提高25%左右的视图合成速率. 也证明了粗糙网络除了可以指导非均匀采样,还可以代替细腻网络的均匀采样部分参与视图合成,而不降低最终的视图质量. 但是,改进后的算法在每个场景上所需的训练时间仍在1天以上,合成一张新视图的时间需要数十秒,还无法做到实时渲染. 因此,继续提升神经网络光场方法的效率仍十分必要. 目前每束光线的采样数量都相等,未来可通过学习的方式自适应地调整每束光线的采样数目,进一步减少总的采样数量,提高视图合成的效率.

参考文献:

[1] Levoy M, Hanrahan P. Light field rendering[C]// Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques-SIGGRAPH '96. New York: ACM Press, 1996: 31-42.

[2] Wood D N, Azuma D I, Aldinger K, et al. Surface light fields for 3D photography[C]// Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques-SIGGRAPH '00. New York: ACM Press, 2000: 287-296.

[3] Lombardi S, Simon T, Saragih J, et al. Neural volumes

[J]. ACM Transactions on Graphics, 2019, 38(4): 1-14.

[4] Mildenhall B, Srinivasan P P, Ortiz-Cayon R, et al. Local light field fusion[J]. ACM Transactions on Graphics, 2019, 38(4): 1-14.

[5] Liu Lingjie, Gu Jiatao, Lin Kyaw Zaw, et al. Neural sparse voxel fields[C]// Advances in Neural Information Processing Systems 33, NeurIPS 2020. Cambridge: The MIT Press, 2020: 15651-15663.

[6] Mildenhall B, Srinivasan P P, Tancik M, et al. NeRF: representing scenes as neural radiance fields for view synthesis [M] // Computer Vision-ECCV 2020. Cham: Springer International Publishing, 2020: 405-421.

[7] Lindell D B, Martel J N P, Wetzstein G. AutoInt: automatic integration for fast neural volume rendering[C]// Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE Press, 2021: 14556-14565.

[8] Reiser C, Peng Songyou, Liao Yiyi, et al. KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs[EB/OL]. (2021-03-25)[2021-04-30]. https://arxiv.org/abs/2103.13744.

[9] Landy M, Movshon J. The plenoptic function and the elements of early vision[C]// Computational Models of Visual Processing. Cambridge: MIT Press, 1991: 3-20.

[10] Rahaman N, Baratin A, Arpit D, et al. On the spectral bias of neural networks[C]// International Conference on Machine Learning. Cambridge: PMLR, 2019: 5301-5310.

[11] Sitzmann V, Zollhöfer M, Wetzstein G. Scene representation networks: continuous 3D-structure-aware neural scene representations[M]. Red Hook: Curran Associates Inc, 2019: 1121-1132.

[12] Schönberger J L, Frahm J M. Structure-from-motion revisited[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE Press, 2016: 4104-4113.

[13] Schönberger J L, Zheng Enliang, Frahm J M, et al. Pixelwise view selection for unstructured multi-view stereo[C]// Computer Vision - ECCV 2016. Cham: Springer International Publishing, 2016: 501-518.

[14] Kingma D, Ba J. Adam: a method for stochastic optimization[EB/OL]. (2014-12-22)[2021-04-30]. https://arxiv.org/abs/1412.6980v8.