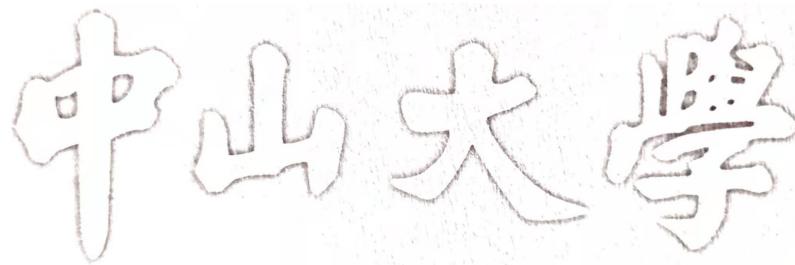


密级: 公开

编号:



工程 硕士专业学位论文

基于神经辐射场的快速新视图合成系统的设计
与实现

Design and Implementation of a Fast Novel View
Synthesis System Based on Neural Radiance Fields

学位申请人: _____

导师姓名及职称: _____

专业、领域名称: 工程硕士(计算机技术)

2021年4月8日

中山大学硕士学位论文

基于神经辐射场的快速新视图合成

系统的设计与实现

Design and Implementation of a
Fast Novel View Synthesis
System Based on Neural
Radiance Fields

专业: 工程硕士(计算机技术)

学位申请人: _____

指导教师: _____

论文答辩委员会主席: _____

成员: _____

二〇二一年四月

原创性及使用授权声明

论文原创性声明内容

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版；有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅；有权将学位论文的内容编入有关数据库进行检索；可以采用复印、缩印或其他方法保存学位论文；可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

保密论文保密期满后，适用本声明。

学位论文作者签名：

日期： 年 月 日

导师签名：

日期： 年 月 日

摘要

新视图合成作为计算机视觉与计算机图形学交叉领域的任务，一直是工业界和学术界研究的经典问题。新视图合成任务是通过给定的单张或者一系列的相机拍摄的三维场景的图像，来合成新视角下的高质量的图像。在很多领域，例如增强现实、虚拟现实、3D 游戏、街景地图等，用户都有想从任意视角下快速观测到三维场景的需求。此外，一个典型的应用场景是自由视点视频，用户可以通过交互控制视点并可以从任意 3D 位置生成动态场景的新视图，在这种情形下渲染速度的快慢直接影响到用户体验。因此，可以在任意视角下快速合成新视图的系统具有重要的应用意义。但目前的新视图合成领域存在着以下的问题：1) 目前市场上还没有直接可应用的面向快速新视图合成的交互系统；2) 现有的新视图合成模型，计算时间成本较高，难以完成 PC 端的交互需求。本文正是为解决上述需求构建了一个基于神经辐射场的快速新视图合成系统。

最近，基于 NeRF 的神经渲染方法在新视图合成任务上表现十分惊艳，是目前最佳的方法。为了获取对渲染图象贡献高的采样点，NeRF 使用了分层采样的方法，通过 coarse 网络的输出估计采样点贡献的大致分布，基于此分布获得较好的采样点送入 fine 网络得到颜色和体密度，最终使用经典体绘制的方法进行数值积分计算出 2D 图像上对应像素的颜色。由于真实场景中自由空间和被遮挡区域对渲染图像没有贡献，因此如何快速获取表面附近的高贡献采样点是一难点。但是，由于大量的采样点需要经过上述两个网络，这使得新视图的渲染过程耗费较长的时间，难以满足本文系统的需求，因此需要对测试时间进行优化，本文正是要通过改进 NeRF 的采样方式来减少一个网络的开销从而解决渲染速度过慢的问题。

值得注意的是，NeRF 具有类似于 PointNet 的逐点网络结构，以 justlookup 为代表的研究工作已经证明了对逐点网络可以去构建查询表来加速网络的前向传播并通过近似特征再学习的策略来维持模型精度不损失。针对上述问题，本文将 NeRF 技术和查询表技术进行有机融合，设计并实现了基于神经辐射场的快速新视图合成系统。具体地，本文的主要工作与贡献如下：1) 使用预训练的 NeRF 网络，构建出一个可以表征在物体内外的查询表，该查询表可以直接为空间内任意一点提供 NeRF 网络中间层的特征，这加快了神经网络的正向传播速度。2) 在缓存了上述查询表的基础上，本文改进了 NeRF 的采样过程，通过查询表找到光线上离物体表面最近的内点，使得 NeRF 可以仅在此点附近进行采样，从而不必再使用两个

摘要

网络，这显著加速了 NeRF，同时这正是本文的核心贡献。3) 本文设计并实现了基于 NeRF 的快速新视图合成的系统，完成了相应的 PC 端的应用，并对该系统进行了详细的需求分析以及架构设计，最后将本文的方法实现并部署到了 PC 端，经过详细地测试，本文方法在合成场景和真实场景下的渲染速度对 NeRF 分别加速 3.2 倍和 3.4 倍，该系统满足实际应用需求。

关键词：神经辐射场，新视图合成，神经渲染，缓存，查询表

ABSTRACT

Novel view synthesis, a cross-field task between computer vision and computer graphics, has always been a classic problem in industry and academia. Novel view synthesis is to synthesize high-quality images from a new view through one or more given images of a 3D scene captured by a camera. In many fields, such as Augmented Reality, Virtual Reality, 3D games, Street View Maps, etc., users have the demand to quickly observe 3D scenes from any view. In addition, a typical application scenario is Free Viewpoint Video, in which users can interactively control the viewpoint and generate a novel view of the dynamic scene from any 3D position. In this case, the rendering speed directly affects the Quality of Experience. Therefore, the system that can quickly synthesize any novel views has important application significance. However, the current novel view synthesis has the following problems: 1) there is no directly applicable interactive system for fast novel view synthesis on the market; 2) the existing novel view synthesis models have high computational time cost, and it is difficult to meet the interaction requirements of PC. In this paper, we construct a fast novel view synthesis system based on neural radiance fields to meet the above requirements.

Recently, the NeRF-based neural rendering method performs amazingly on the task of novel view synthesis, and it is currently the best method. In order to obtain the high-contribution sampling points to the rendered image, NeRF uses the hierarchical volume sampling, estimates the approximate distribution of the contribution of the sampling points through the output of the coarse network, obtains the better sampling points based on this distribution, and then sends them to the fine network to get the color and volume density. Because the free space and the occluded regions in the real scene do not contribute to the rendered image, it is a difficult problem to obtain high-contribution sampling points quickly near the surface. Finally, NeRF uses the classical volume rendering to perform numerical integration to calculate the color of the corresponding pixel on the 2D image. However, since plenty of sampling points need to pass through the above two networks, it takes a long time to render the novel view which is difficult to meet the requirements of the system in this paper. And therefore, we ought to optimize the test time. In this paper, we aim to reduce the overhead of one network by improving the sampling method

ABSTRACT

of NeRF to solve the problem of slow rendering speed.

It is worth noting that NeRF has a point-wise network structure similar to PointNet. Research work represented by justlookup has proved that lookup tables can be built for a point-wise network to accelerate the forward propagation of the network and maintain the accuracy of the model through the strategy of relearning approximate features. For the above problems, we organically integrate NeRF technology and lookup table technology to design and implement a fast novel view synthesis system based on neural radiance fields. Specifically, our main work and contributions are as follows: 1) Using the pre-trained NeRF network, we construct a lookup table that can be characterized inside and outside the object. The lookup table can directly provide the feature map of the NeRF network middle layer at any point in the space, which speeds up the forward propagation speed of the neural network. 2) On the basis of caching the above lookup table, we improve the sampling process of NeRF. The inner point closest to the surface of the object on the light is found through the lookup table, so that NeRF can just sample near this point, eliminating the need to use two networks which significantly accelerates NeRF. And meantime this is the core contribution of this paper. 3) In this paper, we design and implement a fast novel view synthesis system based on NeRF, and complete the corresponding PC application, and then perform a detailed requirements analysis and architecture design for the system. At last, the method in this paper is implemented and deployed to the PC. After detailed testing, the rendering speed of this method in synthetic scenes and real scenes accelerates NeRF by 3.2 times and 3.4 times, respectively, and the system meets actual application requirements.

Keywords: Neural Radiance Fields, Novel View Synthesis, Volume Rendering, Caching, Lookup Tables

目 录

摘 要.....	I
ABSTRACT.....	III
目 录.....	V
第一章 绪论.....	1
1.1 研究背景与意义	1
1.2 国内外研究现状	3
1.3 本文的主要工作与贡献	6
1.4 本文的章节安排	7
第二章 相关技术和理论概述.....	8
2.1 神经三维形状表示法	8
2.2 新视图合成和基于图像的渲染	10
2.3 深度学习发展概述	11
2.4 查询表的加速方法	15
2.5 本章小结	16
第三章 本文提出的方法.....	17
3.1 问题定义	17
3.2 新视图合成的神经辐射场表示	17
3.3 基于神经辐射场的新视图合成加速方法	20
3.4 本章小结	27
第四章 实验部分.....	28
4.1 实验基本设置	28
4.2 实现细节	30
4.3 评测指标	31
4.4 实验结果与分析	33
4.5 本章小结	37
第五章 系统设计与实现.....	38
5.1 系统需求分析	38

目 录

5.2 系统设计	43
5.3 系统实现	43
5.4 本章小结	46
第六章 系统的部署与展示.....	47
6.1 系统运行环境	47
6.2 系统功能测试	49
6.3 本章小结	55
第七章 总结与展望.....	56
7.1 总结	56
7.2 展望	57
参考文献.....	58
在学期间完成的相关学术成果.....	64
致 谢.....	65

第一章 绪论

本章作为绪论主要介绍了神经辐射场中新视图合成加速问题的背景与意义，分析了目前相关技术与算法的发展现状以及相关系统存在的问题，阐述了解决该问题的必要性，然后总结了本文的主要工作与贡献，最后概述了本文的章节结构。

1.1 研究背景与意义

近年来，随着深度学习与计算机视觉的飞速发展，人类的生活愈来愈趋向智能化。在以前，人们必须通过行万里路来了解这个世界，但是现在，可以通过全景地图几乎全方位地去观察一个未知的地方，甚至是通过虚拟现实，增强现实等形式沉浸式地以任意的角度呈现出场景或者物体。新视图合成技术可以支持以上应用场景，可以通过已有的图像去合成新视角下的图像。

新视图合成任务^[1]作为计算机视觉和计算机图形学的交叉领域任务，有着极其广阔的应用前景，一直是学术界和工业界的热门研究问题。新视图合成任务具体是，通过给定的已知视角下观测到的图像，去合成新视角下的图像。图 1-1给出了新视图合成任务的示意图。



图 1-1 新视图合成的示意图^[2]。前三张图是合成的新视图，最后一张图阐明了绿色相机位姿下的图像作为输入，推理红色相机位姿的新视图

根据前期调研，新视图合成技术有着广泛的应用前景，市场需求庞大，急需可快速合成高质量新视图的系统。而在实际应用中，限于应用需求和经济成本等因素的考量，市场上的大多产品仍存在着许多问题，比如无法在任意角度合成高质量图像，另外渲染新视图的时间过长，这些都严重影响了用户体验。而本身，三维场景具有一定复杂性，市面上的很多软件在渲染之前需要进行复杂的三维建模，这很考验硬件的算力和内存。这些都增加了系统的实现难度。

具体地，现有的系统中存在以下一些问题：1) 当前市场上的三维渲染的软件难以根据稀疏视图，在任意视角下渲染出高质量的新视图；2) 目前的三维渲染系统大都需要根据深度信息进行三维建模，这需要使用 RGBD 相机，对硬件设备要求较高；3) 目前市场上还没有直接可应用的面向快速新视图合成的交互系统；4) 现有的新视图合成模型需要在空间内进行大量采样，计算时间成本较高，难以完成 PC 端的快速交互需求。

如果想要把新视图合成技术很好地应用在虚拟现实等技术上，那么必须能够在任意视角下合成几乎与真实场景相同的图像。这在合成场景下是可能实现的，但是在真实场景中，无法获得准确的相机位姿、光照条件难以建模等问题都限制了更好的体验，新视图合成仍面临着许多挑战。

经典的基于图像的渲染 (Image-based Rendering, IBR^[3]) 方法可以完成以上任务，IBR 通常依赖于基于优化的多视图立体 (Multi-View Stereo, MVS) 方法，以此重构场景的几何结构。然而，如果只有少量观测图像可用，场景包含依赖视图的影响或者新视角下大部分没有被观测的图像覆盖到，那么 IBR 可能失效会导致出现鬼影或者空洞等效果。此外该方法需要对三维场景进行精细的三维重建，这会带来极大的计算开销，因此传统的基于图像的渲染方法难以应用到实际场景中。随着深度学习的发展，基于深度学习的神经渲染方法开始在新视图合成任务上展现出其惊人的渲染效果。2020 年，Mildenhall 等人^[4] 提出了基于神经辐射场 (Neural Radiance Fields, NeRF) 的方法去合成新视图，这使得神经渲染一度成为最优秀的的新视图合成方法。NeRF 基于给定的稀疏图像，仅使用较为简单的全连接网络去隐式表达场景信息，最后使用体绘制的方法合成新视图。如果有一个系统能够使用基于 NeRF 的方法快速渲染出高质量的新视图，那么将极大地提升用户体验。因此，在这样的需求下，本文致力于将具有最佳渲染效果的 NeRF 技术设计并实现成完整的新视图合成系统，并在不损失精度的情况下对其渲染过程进行加速。

NeRF 是目前新视图合成任务中的渲染效果最佳的方法，但是由于使用体绘制进行渲染，要求必须在空间内进行大规模采样，这一方面会带来巨大的时间开销，另一方面采样点的位置的准确性将决定了神经辐射场拟合的准确性。

本文主要关注的是渲染时间长这一问题。NeRF 为了获取对渲染图象贡献较高的采样点，同时优化两个神经网络，使用 coarse 网络的输出去估计采样点的分布情况，然后基于此分布进行二次采样并通过 fine 网络预测的颜色和体密度进行数值积分计算出 2D 图像对应像素的 RGB 值，这带来了极大的时间开销。同时为了拟合更准确的神经辐射场，NeRF 使用了较深的神经网络，这也增加了推理新视图过程的时间成本。以上正是本文需要加速的地方。

在这样的背景下，本文研究了基于神经辐射场的新视图合成的相关技术，认为基于深度学习的神经渲染方法存在着提升空间，并对现有方法 NeRF 进行了优化，设计并训练了可以快速合成高质量新视图的神经渲染模型。渲染效果受数据集的影响，一般来说合成场景下质量会相对好些，因为真实场景下问题比较多，比如相机标定不严格、相机位姿估计不准，照片畸变严重，光照条件难以建模等等。

最终，本文结合 NeRF 的技术以及查询表缓存加速的相关技术实现了快速合成新视图的神经辐射场渲染具体方法，并设计和实现了一个快速新视图合成的系统，完成了 PC 客户端的相关应用。与现有产品相比，该系统可以基于给定的稀疏图像，快速渲染出任意新视角下的图像，提供几乎和 NeRF 质量相同的新视图。

本课题来自于 PixTalks 公司的实际应用需求。

1.2 国内外研究现状

1.2.1 基于图像的神经渲染方法研究现状

在计算机图形和计算机视觉中，基于图像的渲染方法通常是依靠场景的一组二维图像来生成三维模型，然后渲染该场景的一些新视图。一般基于图像的渲染 (Image-based Rendering, IBR) 方法根据是否使用几何信息分成两类。几何信息用于将图像内容从捕获的图像重新投影到新的目标图像域。在目标图像域中，源图像的投影被混合以合成最终图像。这种简化的过程仅能为具有精确几何结构且具有捕获足够多视图的物体提供较为准确的结果。但是，由于依赖于视图的影响，以及不完善的几何信息或太少的源图像，可能会出现诸如重影，模糊，孔洞或接缝之类的伪影。

为了解决上述问题，将经典的基于图像渲染的方法与现如今最火的深度学习网络的结合成基于 IBR 的神经渲染方法，这极大地提升了 IBR 的渲染效果。Deep-Blending^[5] 提出了一种广义网络来预测投影源图像的混合权重，以便在目标图像空间进行合成。DeepBlending 在室内场景中显示了惊人的效果，比经典的 IBR 方法具有更少的混合伪影，具体见图 1-2。事实上，DeepBlending 是将 COLMAP^[6] 和商用软件 Reality Capture 进行了结合，以此获取更为精细的几何结构信息。COLMAP

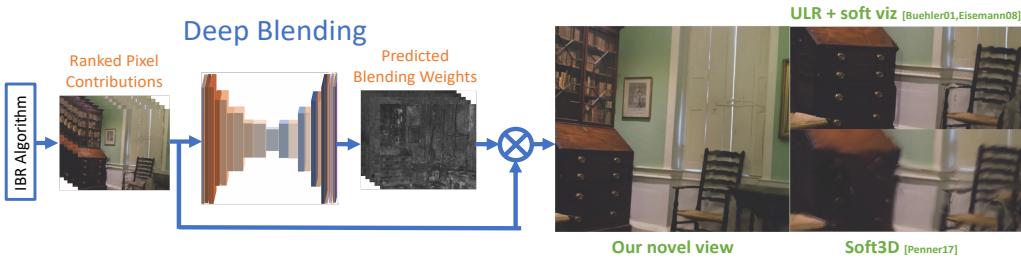


图 1-2 DeepBlending^[5] 的框架结构，渲染效果较其他方法减少了伪影

基于块匹配的多视图几何算法进行稠密重建，这能够非常好地重建场景中的细节，但是与此同时在纹理信息缺失的地方也会出现空洞。Reality Capture 的结果相对完整，但是对于场景级别的区域重建的较差。因此，上述两种方法是互补的，将它们进行结合可以获得非较为准确的几何模型信息，DeepBlending 就是使用上述几何信息，借助神经网络来学习如何将不一致的观测图像信息进行混合，最终确实获得了非常惊艳的效果。但是，该方法过于依赖 COLMAP 三维重建出的几何信息，当几何信息较少时渲染质量会严重下降并伴随伪影。此外，精细的三维重建会带来非常大的计算开销，这在实际中难以使用。因此这种经典的基于 IBR 的方法无法满足本系统的速度和质量需求。

1.2.2 基于 NeRF 的新视图合成研究现状

类似 DeepSDF^[7]，NeRF^[4] 借助深度学习的隐式表达方法，将三维几何场景中的光照和几何信息隐式表达在神经网络参数中，最后利用立体渲染的方法进行渲染。NeRF 算是基于图像的神经渲染方法中最具代表的工作了。NeRF 本身不需要任何的显式的几何信息，仅使用真实场景的 RGB 图像作为监督信号，使用深度神经网络自动推理相关几何以及光照信息。NeRF 凭借着其网络简单，渲染效果惊艳的特性，备受学术界和工业界的关注。自从 NeRF 的出现，基于神经辐射场的新视图合成相关工作开始大量涌现。

NeRF++^[8] 首次指出 NeRF 具有形状光线的歧义性问题，即 NeRF 在一个场景下训练好的网络，其对应的空间表示可能是错的，但是仍能在训练集上渲染出正确的结果。

Yen-Chen 等人^[9] 基于位姿估计方法给 NeRF 提供了新的改进思路，推出了新工作 iNeRF。NeRF 在处理真是数据集的时候，位姿是通过 COLMAP^[6] 进行计算的，然而这并不能获得绝对精准的位姿，也即送入网络的采样点是不准确的，那么也无法准确拟合真是场景的神经辐射场。如图 1-3所示的是 iNeRF 的网络框架，注意到，颜色的均方误差对坐标是可微的，而坐标对相机位姿也是可微的，因此，可

以在训练神经辐射场的同时对位姿进行优化，具体是采用神经网络的反向传播算法进行优化的。此外，为了更有效地优化位姿，iNeRF 使用了 ROI 采样方式，即对感兴趣的区域进行采样优化，这样能显著缓解直接采样一张图片带来的内存开销大的问题。iNeRF 通过位姿矫正，最终在真实场景下显著提升了 NeRF 的渲染质量。不过，由于训练时每一张图都要单独优化位姿，而渲染时相机位姿仍然是不准确的，因此本系统暂不考虑位姿的问题。

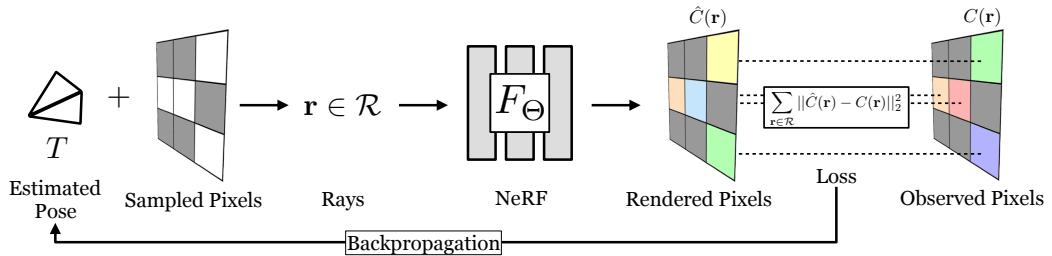


图 1-3 iNeRF^[9] 的网络框架示意图

上述方法都还是基于多层感知机 (MLP, Multilayer Perceptron) 训练的。下面介绍一些对 NeRF 网络上进行的一些改进的工作。PixelNeRF^[10] 使用卷积神经网络 (Convolutional Neural Networks, CNN) 提取的特征，仅需要少量图片作为输入就可以合成新视角下的图像，并有一定的泛化能力。GRF^[11] 使用 CNN 提取包含光线属性的特征，并使用 attention 机制将采样点在不同视角下的特征进行聚合，极大地提升了渲染质量。GRAF^[12] 使用对抗生成网络 (Generative Adversarial Networks, GAN) 对 NeRF 进行改进，使之具有较强的泛化能力。但是，以上这些基于网络的改进方法都使得网络变得过于复杂，有些违背 NeRF 网络简洁的初衷，会使得训练和渲染的时间过长，这不能满足用户对于快速合成新视图的需求，几乎无法应用到实际的系统中。

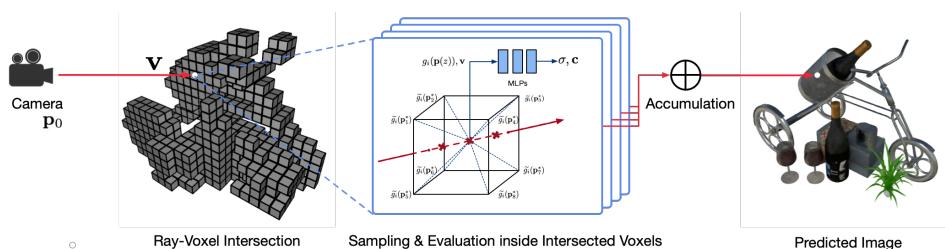


图 1-4 NSVF^[13] 的网络框架示意图

下面介绍一些基于 NeRF 的加速方法。由于 NeRF 需要在相机光线上进行大量的采样，这些采样点都要经过网络，而 NeRF 自身又使用了两个网络进行训练，因

此 NeRF 的渲染是很慢的。如图 1-4, 为了给 NeRF 加速, NSVF^[13] 首先提出将空间划分为若干体素, 使用自我裁剪的方法去掉体密度较小的体素, 渲染的时候只需找到相机光线第一个穿过的体素, 根据该体素八个顶点对应的特征进行插值编码并送入下游 MLP 预测出颜色和体密度, 这避免了大量的采样计算, 显著加速了新视图合成的过程。但是, NSVF 仅适用于简单的三维物体, 在复杂的真实场景下渲染新视图的质量较差, 甚至会出现鬼影, 因此并不适合在本系统上使用。Lindell 等人^[14] 提出了一种利用自动积分来进行渲染加速的方法 AutoInt。AutoInt 是通过隐式网络学习到闭式解求积分, 从而取代了 NeRF 的数值积分, 最终能为 NeRF 渲染过程加速 10 倍。不过, 就 PSNR 来看, AutoInt 在合成数据集 Synthetic-NeRF 上平均比 NeRF 低 5.4 dB, 这是一种重视加速但牺牲了很多质量的方法, 并没有真正做到在不失精度的情况下对渲染进行加速。上述介绍的加速方法未能保证不损失精度情况下对 NeRF 进行加速, 不符合本系统的需求, 因此本文最终依旧使用最经典应用场景最丰富的 NeRF 技术并对其进行无精度损失的加速, 最终完成整个快速新视图合成系统的构建。

1.3 本文的主要工作与贡献

本文将 NeRF 技术和查询表技术进行有机融合, 设计并实现了基于神经辐射场的快速新视图合成系统。本文根据 NeRF 的特性提出了基于神经辐射场进行加速合成新视图的框架, 将其命名为 F-NeRF, 通过 NeRF 预测的体密度有效地提取到物体的几何结构并缓存到查询表中, 这一方面辅助了获取物体表面附近的对计算颜色有高贡献的采样点, 另一方面减少了一个网络的开销, 同时也简化了网络, 这显著加速了渲染的过程。最后, 本文将这 F-NeRF 这一方法有效地部署到 PC 端, 将其实现成可交互的系统。

具体地, 本文的主要工作与贡献如下:

- 1) 本文基于 NeRF 预训练的网络预测的体密度信息构建出一个可以表征在物体内外的查询表结构, 这一方面直接减少了查询表的尺寸, 另一方面该查询表可以直接为空间内任意一点提供 NeRF 网络中间层的特征, 这加快了神经网络的正向传播速度。
- 2) 在缓存了上述查询表的基础上, 本文改进了 NeRF 的采样过程, 通过查询表找到光线上离物体表面最近的内点, 使得 NeRF 可以仅在此点附近进行采样, 从而不必再使用两个网络, 这显著加速了 NeRF, 同时这正是本文的核心贡献。
- 3) 为了补偿查询表分辨率不够带来的精度损失, 本文对 F-NeRF 进行再训练, 将

渲染质量提升到原 NeRF 的水平。

- 4) 本文在公开的数据集上做了大量对比验证实验。在公开合成数据集 Synthetic-NeRF 和真实场景数据集 LLFF-NeRF 上, 与 NeRF 相比, 我们提出的 F-NeRF 方法分别加速 3.2 倍和 3.4 倍。
- 5) 本文设计并实现了基于 NeRF 的快速新视图合成的系统, 完成了相应的 PC 端的应用, 并对该系统进行了详细的需求分析以及架构设计, 最后将本文的方法实现并部署到了 PC 端, 经过详细地测试, 该系统满足实际应用需求。

1.4 本文的章节安排

第一章是绪论, 主要阐述了本文研究工作的背景与意义, 指出了本文研究问题的难点, 并介绍了本文的主要研究工作和贡献, 对国内外相关工作的研究现状的充分调研与分析, 最后对本文的组织架构进行安排。

第二章是相关理论和技术, 主要介绍了基于神经辐射场的快速新视图合成系统应用的核心技术和算法。

第三章是本文的主要工作, 主要介绍了本文方法的问题定义, 本文方法的预备工作, 然后详述了我们改进的基于神经辐射场的新视图合成加速方法。

第四章是本文的实验部分。主要阐述了实验的基本设置, 包含开发环境、数据集等等, 其次介绍了实验实现的具体细节以及相关评测指标。最后通过大量对比实验来验证本文提出的方法的有效性, 同时对一些实验结果进行分析和讨论。

第五章是系统设计与实现, 主要介绍了将本文提出的 F-NeRF 方法应用到系统上的具体实现与架构设计。详细阐述了本系统的需求分析, 包含主要用例分析, 以及具体的系统架构设计, 最后介绍了将本系统的模型移植到 C++ 代码上的方法。

第六章是系统的部署与展示, 主要介绍了本文系统的开发与运行环境, 对经典的测试用例进行了详尽的步骤说明, 根据用例步骤完成了功能测试。从测试结果看, 系统各项功能均能正常运行, 合成新视图速度快且质量高。

第七章为总结和展望部分。总结了本文的主要工作和贡献, 并指出工作中的不足之处。提出在未来研究过程中的可能的研究方向, 并对可继续完善的方面进行展望。

第二章 相关技术和理论概述

本文主要研究的内容是神经辐射场在新视图合成上的加速，因此本章首先主要介绍神经三维形状表示法，新视图合成和基于图像的渲染方法。随着深度学习的发展，当今效果最佳的新视图合成算法都是基于深度学习的，因此接下来会概述深度学习的原理和发展状况，并重要介绍基于深度学习的神经辐射场的立体渲染方法。最后介绍一下查询表的概念，以及查询表在逐点网络上的加速应用，为本文方法的展开做好铺垫。

2.1 神经三维形状表示法

三维物体的形状表示有多种方法，传统的基于数据驱动的三维学习表示方法可以总体上分为三类：基于点，基于网格，基于体素的方法。此外，还有最新的基于深度学习的三维形状表示方法，下面对以上四种表示方法分别进行阐述。

2.1.1 基于点的表示方法

基于点的方法一般指的是点云（如 2-1 所示的第一个）这种表示，点云是指空间中一组点的集合，每一个点都包含 X, Y, Z 三个坐标。它是一种轻量级的三维表示方式，与匹配许多传感器（例如雷达，深度相机）提供的原始数据非常匹配，因此非常适合应用在三维学习技术上。例如 PointNet^[15-16] 这篇非常经典的点云工作，使用最大池化操作来提取全局特征，之后这项技术被广泛用作点生成网络^[17-18] 的编码器。此后点云学习方面又涌入了大量的 PointNet 风格的相关工作。然而，点云这种表示方法有着很明确的限制，它们不适用于描述拓扑结构，同时也不适合用以生成紧致的表面。

2.1.2 基于网格的表示方法

许多方法使用预定义模板的网格，它们表征了一系列相似形状的物体，例如可变形的人体部位，其中一些模型^[19] 展示了高保真的形状生成结果。其他的最近工作^[20] 使用多边形立方体映射^[21] 对形状进行优化。虽然模板网格的使用很方便，并且自然地提供了三维对应，但它只能对具有固定网格拓扑的形状进行建模。其他基于网格的方法使用已有的^[22-23] 或者学习的^[24-25] 参数化技术通过二维平面来描述三维曲面。这种表示的质量取决于参数化算法，这些算法通常对输入网格质

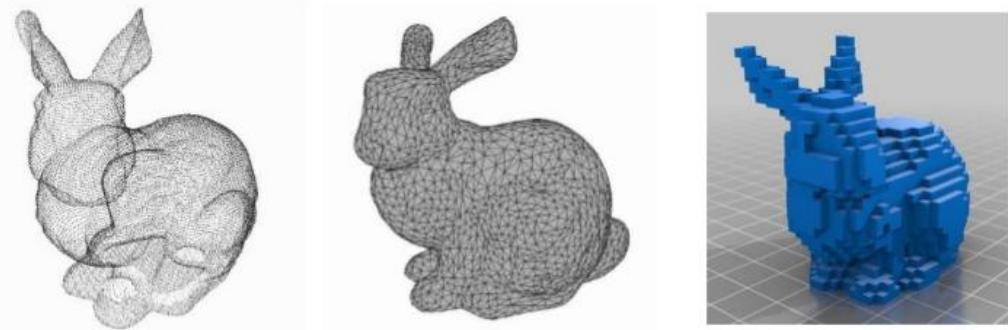


图 2-1 三种传统的三维表示方式（点云，网格，体素）

量和切割策略非常敏感。为了解决这个问题，最近的数据驱动方法^[17]学习深度网络的参数化任务。然而，他们报告说需要多个平面来描述复杂的拓扑，但是生成的曲面片没有缝合，即生成的形状没有闭合。为了生成闭合网格，可以使用球体参数化^[24-25]，但生成的形状仅限于拓扑球体。其他与网格学习相关的工作建议对网格^[26-27]或一般图^[28]使用新的卷积和池化运算。网格表示三维物体的直观感受如 2-1 所示。

2.1.3 基于体素的表示方法

在三维计算机图形学上，体素代表了三维空间中规则格点所对应的值。与二维位图中的像素一样，体素本身通常不使用其值显式编码其位置（即坐标）。取而代之的是，渲染系统根据其相对于其他体素的位置（即其在构成单个体积图像的数据结构中的位置）来推断体素的位置。体素算是像素由二维空间直接推广到三维空间，有着不少共性。基于体素的学习方法中最直接的变体就是占据栅格。但是由于计算和内存开销呈三次方的增长趋势，因此当前的方法只能处理较低分辨率的情况。因此，基于体素的方法^[29-30]无法保留精细的形状细节，由如 2-1 所示的第三个图就能明确这一点，此外，体素在视觉上显示与高保真度形状有显著差异，因为渲染时法线不平滑。基于八叉树^[31-33]的方法一定程度上减轻了稠密体素方法的计算和内存限制，但是提升后的分辨率依然没有产生很好的视觉响应。

2.1.4 基于深度学习的表示方法

计算机视觉中最近一个有前景的方向是用多层感知机 MLP 的权值编码对象和场景，MLP 直接从三维空间位置映射到形状的隐式表示，例如该位置的有符号距离^[34]，如图 2-2 所示是有符号距离函数 (Signed Distance Function, SDF) 在三维表示上的应用。然而，这些方法到目前为止还无法再现具有复杂几何体的真实场景，其保真度与使用离散表示（如三角形网格或体素网格）表示场景的技术相同。

最近的工作通过优化将 xyz 坐标映射到 SDF 函数^[7,35-36]或占据域^[37-38]的深

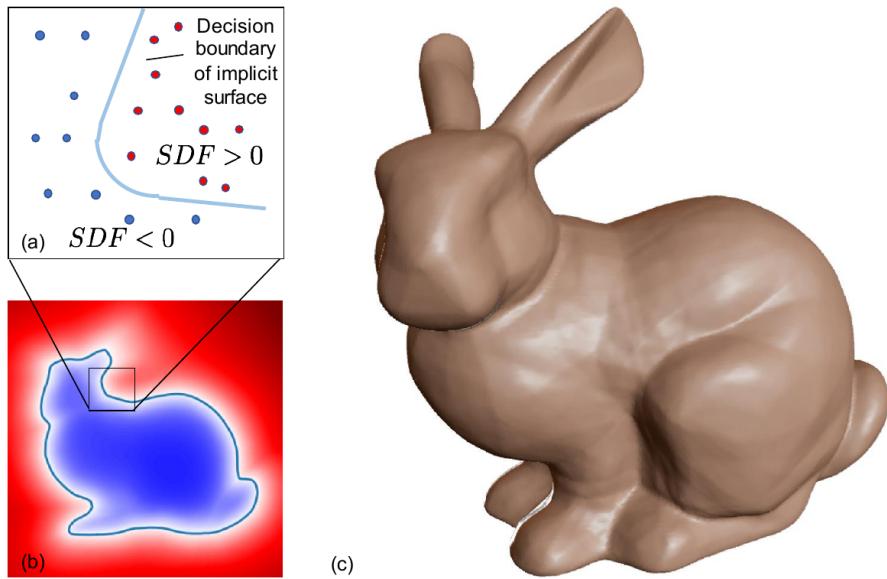
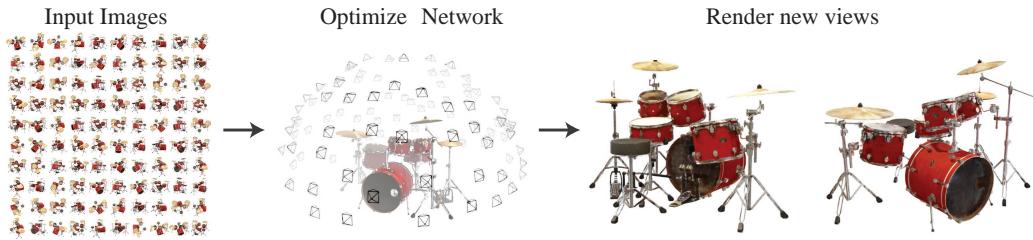


图 2-2 SDF 函数表示法应用于斯坦福兔子，图片来自于 DeepSDF^[7]

度网络，研究了连续三维形状作为水平集的这种隐式表示。然而，这些模型受到三维几何体 ground truth 的需求的限制，这些几何体通常是从合成的三维形状数据集（例如如 ShapeNet^[39]）获得的。随后的工作通过构造可微的渲染函数，使得神经隐式形状表示仅使用 2D 图像进行优化，从而放宽了对三维形状 ground truth 的要求。Niemeyer 等人^[40] 将曲面表示为 3D 占据场，并使用数值方法找到每条光线的曲面交点，然后使用隐式微分计算精确导数。每个光线相交位置都作为神经 3D 纹理场的输入，该纹理场预测该点的颜色。Sitzmann 等人^[41] 使用了一种不太直接的神经 3D 表示法，只需在每个连续的 3D 坐标上输出一个特征向量和 RGB 颜色，并提出了一种可微绘制函数，该函数由一个沿每条射线行进的递归神经网络组成，以确定曲面的位置。

2.2 新视图合成和基于图像的渲染

新视图合成在计算机视觉与计算机图形学这样的交叉领域上是一个长期的问题。所谓新视图合成指的是给定一系列（或者单张）已知相机位姿的视图，由这些视图合成出未知位姿的视图。新视图合成的过程如图 2-3 所示。对于给定稀疏视角采样的新视角合成问题，计算机视觉和图形学社区在从观测到的图像中预测传统几何表示的问题上取得了很多进展。一类流行的方法是使用具有要么漫反射^[42] 的，要么依赖视角^[43-44] 的外观的那些 mesh-based（基于网格）的场景表示。可微分的光栅化器^[45-48] 或路径跟踪器^[49-50] 可以直接通过使用梯度下降法来优化再现一组输入图像的网格表示。然而，基于图像重投影的网格梯度下降的优化是十分

图 2-3 新视角合成的过程^[4]

困难的，可能是由于局部极小值或 loss 的调整不佳所致。此外，这种策略^[49] 要求在优化之前提供具有固定拓扑的模板网格作为初始化，这通常对于不受约束的真实场景是不可用的。

另一类方法使用体积表示法来解决从一组输入 RGB 图像中进行高质量逼真的视图合成的任务。体积表示法是能够真实地表达复杂的形状和物体，非常适合基于梯度的优化，比基于网格的方法产生的视觉干扰更少。早期的体积方法^[51-52] 使用观察到的图像直接为体素网格着色。最近，几种方法^[53-56] 使用了多个场景的大型数据集来训练深层网络，这些深层网络从一组输入图像中预测采样的体积表示，然后使用其中一种 alpha 合成^[57] 或沿光线学习的合成，以在测试时渲染新视图。其他工作需要针对每个特定场景优化了卷积神经网络和采样体素网格的组合网络，这样 CNN 可以补偿来自低分辨率体素网格^[58] 的离散化伪像或允许预测的体素网格根据输入时间或动画控制而变化。尽管这些体积化的技术在新视图合成中取得了令人印象深刻的效果，但是由于它们是离散采样的，它们在时间和空间上的复杂性从根本上限制了它们缩放到高分辨率图像的能力，渲染高分辨率的图像需要 3D 空间的更精细的采样。NeRF 通过在全连接的深度神经网络的参数范围内编码连续的体积来规避此问题，这不仅比以前的体积方法产生了更高质量的渲染，而且只需要这些采样体积表示的存储成本的一小部分。

2.3 深度学习发展概述

深度学习 (Deep Learning, DL) 是机器学习的一个分支，是一种以人工神经网络 (如图 2-4) 为架构进行表征学习的方式。深度学习有着非常悠久的历史，最早可以追溯到 20 世纪 40 年代^[59]。学习可以是有监督的、无监督的，也可以是半监督的。

深度学习发展到现在，得益于 GPU 性能的巨大提升，已有诸如深度神经网络 (Deep Neural Network, DNN)，深度置信网络^[60] (Deep Brief Network, DBN)，循环神经网络^[61] (Recurrent Neural Network, RNN)，卷积神经网络^[62] (Convolutional

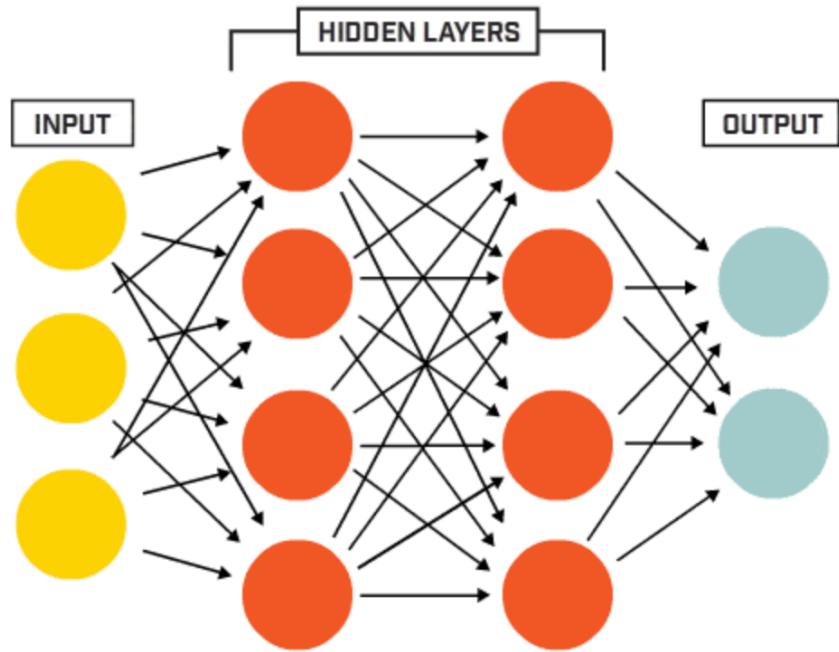


图 2-4 神经网络示意图

Neural Networks, CNN) 等深度学习网络模型已经服务于人工智能的各种领域。

由于本文主要涉及计算机视觉和计算机图形学的知识，因此接下来两小节会介绍这些领域最常用的两个网络架构，卷积神经网络和残差网络 Resnet。

2.3.1 卷积神经网络

要具体阐述卷积神经网络，就不得不先提到 MLP，如图 2-4，一个最简单的 MLP 由三部分组成，包含基本的输入输出还有隐藏层。其中，隐藏层经常采用 sigmoid 和 tanh 作为激活函数。若不使用激活函数，即使是使用多层网络进行训练，实际上只相当于一层，只能进行简单的线性拟合，而有了激活函数网络则能够进行复杂的非线性拟合。不过在输入过大或者过小的时候，上述两种激活函数会出现梯度接近于 0 的情况，此时梯度更新幅度较小甚至是消失，神经网络的训练过程停滞。与 tanh 相比，sigmoid 还有个非常不利于学习的问题，那就是 sigmoid 函数的值域是 $[0, 1]$ ，也因此输出的均值不会为 0，这非常不利于下一层的学习。为了解决上述问题，Glorot 等人^[63] 提出了 ReLU 函数，解决了在输入为正数时梯度消失的问题，此外由于计算比 sigmoid 和 tanh 简单，前向传播和反向传播也都相应更快。

对于正向传播， l 层的输出和 $l - 1$ 层的关系始终可以由下列公式来表达，而

与网络深度无关。

$$a^l = \sigma(z^l) = \sigma(f(a^{l-1})), \quad (2-1)$$

其中 $z^l = f(a^{l-1})$, f 表示第 l 层网络对应的映射函数, a^{l-1} 是前一层的输出 (或第 l 层的输入), 此外 σ 是激活函数。而反向传播则可用链式法则来进行每一层的求导, 计算出梯度。

但是对于图像这种数据来说, 全连接网络有着很大的问题, 一方面, 比如即便是 $256 \times 256 \times 3$ 这样的小图像也需要数十万的参数, 这么多参数会使得网络的训练难以进行下去; 另一方面, 图像数据不像其他任务中的数据, 图像本身具有一定的高级语义, 为了更好处理地处理像图像这样的二维数据, 卷积神经网络应运而生。CNN 是一种前馈神经网络, 除了全连接层, 它还拥有着独有的卷积层, 此外也会使用下采样的池化层。这一结构使得 CNN 对二维结构的数据 (例如图像) 进行很好的处理, 当然训练过程也可以使用反向传播算法^[61]。由于存在着共享参数这一特性, 卷积神经网络需要学习的参数比 MLP 更少, 这使之成为深度学习领域影响力颇深的一种结构。

1. 卷积层

实际上, 卷积层是对输入进行特征提取的一种结构。卷积层一般会包含多个神经元, 或者叫做卷积核, 组成卷积核的每一个元素都会包含一个权重和一个偏移量 (Bias Vector)。卷积核在工作的时候会有规则的扫过输入的特征, 对卷积核大小范围内的特征做矩阵乘法并求和, 最后再叠加上偏移量。

对一张二维图像 X , 卷积层的输出可表示为

$$y_{ij} = \sum_{u=1}^m \sum_{v=1}^n w_{uv} * x_{i-u+1, j-v+1}, \quad (2-2)$$

其中 x 表示输入图像 $X \in R^{M \times N}$ 的一个像素大小, w 是卷积核 $W \in R^{m \times n}$ 的参数, 一般地 $m \ll M$, $n \ll N$ 。二维卷积的示意图如图 2-5

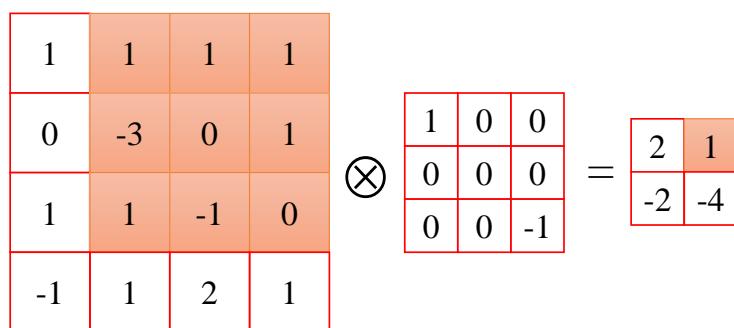


图 2-5 二维卷积过程

2. 池化层

池化层可以理解为下采样，减少输出特征的尺寸，能扩大感受野还能减少参数。最大池化和平均池化是最常见的两种方法。一般卷积层后会配合一个池化层，能进一步提取到有效的特征，保留主要特征并减少参数。

3. 全连接层

无论说是一般的深度学习网络还是 CNN 这样稍微复杂的网络，全连接层都是关键所在。对于 MLP 这样的网络，那么就只有全连接层，可见全连接层的重要性。而对于卷积神经网络而言，全连接层就是将卷积层和池化层提取到的特征整合并输出预测结果。

卷积层的提出极大地促进了计算机视觉领域的发展，AlexNet^[64] (5 层卷积层、3 层全连接层) 在 2012 年的 ImageNet 视觉识别挑战赛中斩获冠军，可见 CNN 对于图像这样高级语义任务的巨大能力。此后，VGG^[65] 网络改进了 AlexNet 的网络，使得整个网络更深 (19 层)，这使他成为了 2014 年 ImageNet 挑战赛的第一名。随着 VGG 的提出，人们开始倾向把网络加深，期望获得更好的识别效果。但实际上随着网络的逐渐加深，更深的网络难以训练，出现了因网络太深导致模型退化的问题，这是由于层数越深，训练误差越大，这与深度神经网络的初衷是背道而驰的，也不符合神经网络拟合任意复杂函数的理论。

但是众所周知的是，深度神经网络并没有消失在人们的视野里，这是由于残差网络 Resnet 的出现，这会在下一节进行具体介绍。

2.3.2 残差网络 Resnet

Kaiming He 等人借鉴了 Highway Network^[66] 的跨层连接思想，并提出了赫赫有名的残差网络 Resnet^[67]，较好地解决了当神经网络极深时模型失效无法训练的问题，并且在当时的 ImageNet 大赛上获得了第一名的好成绩。本文使用的神经辐射场的网络是基于残差模块设计的，因此接下来详细介绍残差网络的设计想法。

如图 2-6 所示，经典的残差网络由多个残差模块组成，因此可以根据需求自行设计网络的深度。残差网络核心思想是使用了 shortcut 这一操作，即将输入传到输出作为输出的一部分，用公式表达就是 $H(x) = F(x) + x$ ，当 $H(x) = x$ 时为恒等映射，这里的 $F(x)$ 就是需要学习的残差。这样的设计原理是，当深度学习网络极深的时候，由于训练误差较大，无法提取出更抽象的特征，那么此时有了这样的残差结构，就可以学习到一个恒等映射，至少还能保留前面提取出的特征。虽然学习一个恒等映射不是非常容易，不过由于 shortcut 这一操作使得只需让残差往 0 的方向去学习即可，这样也同时有效避免了梯度消失的问题。resnet 的出现是具有划时代的意义的，它解决了极深的神经网络难以训练的问题，通过有效地学习到

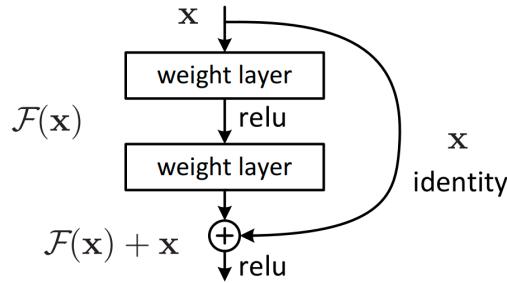


图 2-6 典型的残差结构

恒等映射来保证当网络越来越深的时候所导致的模型退化性能衰退的情况。

本文的基础网络架构就是引入了 skip connection (或者是上述的 shortcut) 的操作来将输入的特征往网络深处传送，以此来训练出一个高质量的深度学习网络，这样才能学习到细粒度的连续的神经辐射场。

2.4 查询表的加速方法

在计算机科学中，查询表 (Lookup Table) 一般是一种简单的数组结构，通过简单的数组索引操作替代一些复杂的计算。一般来说，查询表的加速是很可观的，因为直接从内存中提取到相关数据往往要比复杂的计算快很多。

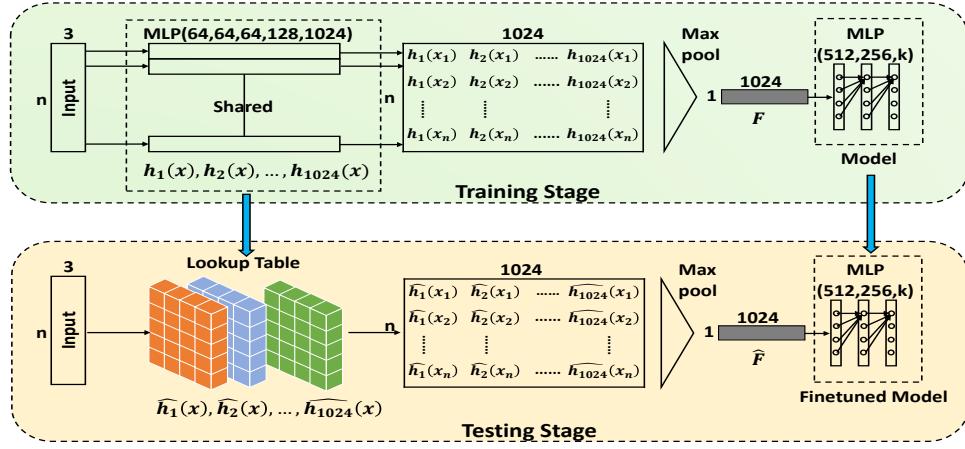
一个典型的应用场景是自然对数表，在一些应用中，直接计算自然对数值是难以接受的，为了解决这一问题，可以提前将一些常用值的自然对数缓存在查询表中，在计算未知的对数值的时候，可以通过直接查询临近值的对数值而避免了复杂的数学计算，这大大加快了计算效率。

在一些细粒度的任务中，往往需要获得较高精度的查询结果，这时候就需要进行一些折衷，比如在使用查询表的同时结合插值的方法，通过略微增加计算量的方式增加了精度，同时也一定程度减少了查询表的尺寸。

值得注意的是，虽然查询表的加速效率一般很高，但是通过查询表替换的计算方式过于简单，那么可能会出现从内存中提取数值的速度低于直接计算的速度，这样就得不偿失。

查询表实质是一种通过空间换时间的方式，但是实际应用中内存是有限的，因此必须要考虑查询表尺寸的问题。

如图2-7所示，以 justlookup^[68] 为代表的工作拓宽了查询表的使用，justlookup 已经证明了对逐点网络可以去构建查询表来加速网络的前向传播并通过近似特征再学习的策略来维持模型精度不损失，这一思想也可以应用到本文的模型加速上，通过查询表缓存网络参数，以此替换掉一些全连接层，这样可以少经过一些网

图 2-7 查询表在逐点网络上的应用^[68]

络而直接通过查询表查询到中间层网络特征，从而加速前向传播。查询表技术在本文的具体应用详见下一章。

2.5 本章小结

本章主要阐述了与本文系统相关的技术和理论方法，包含用于描述 3D 形状的一些表示方法，以及与本文任务最相关的新视图合成和基于图像的渲染，还有关于深度学习的一些基础概念，这些都构成了本文方法的基础。最后介绍了关于查询表的加速方法，以及将其应用到含有逐点网络的前向传播上的经典方法，这一思路也成为了本文方法的根基，将在下一章节详细阐述本文的方法。

第三章 本文提出的方法

上一章对本文研究相关的理论和技术进行了概述。本文的研究始终围绕着“基于神经辐射场的新视图合成加速”这一框架展开。本章主要详细介绍本文的方法论部分，包含问题的定义，NeRF 的相关预备理论，以及本文提出的对 NeRF 加速的具体方法，最后是对本章的总结。

3.1 问题定义

本节将简要阐述本文所研究问题的定义。首先本文的研究目标是基于移动端相机采集的物体的不同视角的图像，快速渲染推理出新视角下的图像。要达到上述目标需要解决 NeRF 网络中如下的几个问题：

1. 由于新视图合成是细粒度的渲染问题，NeRF 为了获取对渲染图象贡献较高的采样点，在训练和推理的过程中使用 coarse 网络和 fine 网络，通过 coarse 网络的输出来估计体密度随深度的变化分布，然后基于此分布进行二次采样并通过 fine 网络预测的颜色和体密度进行数值积分计算出 2D 图像对应像素的 RGB 值，这样两个网络的前向传播带来的开销是比较大的。
2. NeRF 为训练更准确的神经辐射场，使用了深度学习网络，该网络本质上是逐点网络，受 justlookup^[68] 启发，同样可以使用查询表缓存其逐点网络的中间特征，对合成新视图的过程进行加速。

3.2 新视图合成的神经辐射场表示

新视图合成的目的是基于已有的视角的图像，推理未知视角下的图像，现有的基于神经辐射场的体绘制方法已经可以合成高质量的新视角图像。本节详细介绍神经辐射场 NeRF 的主要方法，包含神经辐射场的场景表示，基于神经辐射场的立体渲染，神经辐射场的优化三部分，由 NeRF 的方法论给本文提供一定的问题来源和理论支撑。

3.2.1 神经辐射场的场景表示

NeRF 的整体框架是使用 5D 的向量值函数来表示连续的场景，其中，输入是一个 3D 坐标 $\mathbf{x} = (x, y, z)$ 和 2D 的视角方向 (θ, ϕ) ，输出是发出的颜色 $\mathbf{c} = (r, g, b)$

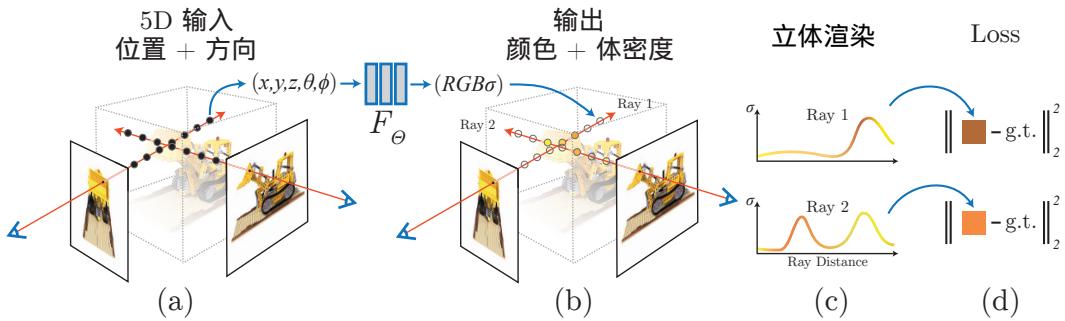


图 3-1 神经辐射场的场景表示和可微的渲染过程概况^[4]。(a) 沿着相机光线采样 5D 坐标 (位置和观测方向); (b) 将 (a) 中位置嵌入 MLP 以生成颜色和体密度; (c) 将 (b) 中的这些输出使用立体渲染的方法合成相应的图像; (d) 此渲染过程是可微的, 因此可以通过最小化合成的和 ground truth 观测图像之间的残差来优化场景表示

和体密度 σ 。实际上, 方向被表示为三维直角坐标系中的单位向量 \mathbf{d} 。NeRF 通过 MLP 网络 $F_\Theta : (\mathbf{x}, \mathbf{d}) \Rightarrow (\mathbf{c}, \sigma)$ 来估计这个连续的 5D 场景表示, 并优化网络权重 Θ , 建立起 5D 坐标到相应的体密度以及相机光线方向发出颜色的映射, 这是神经辐射场最基本的模型。下面将详细介绍基于神经辐射场的立体渲染的方法。

3.2.2 基于神经辐射场的立体渲染

NeRF 的 5D 神经辐射场将场景表示为空间内任意一点的体密度和定向发出的辐射, 并且使用经典体积渲染的原理来渲染穿过场景的任何光线的颜色。体积密度 $\sigma(\mathbf{x})$ 可以解释为一条光线在 \mathbf{x} 位置处终止于无穷小粒子的概率。相机光线 $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ 在近远界限分别为 t_n 和 t_f 时所对应的颜色的期望为:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), \mathbf{d}) dt, \text{ 其中 } T(t) = \exp \left(- \int_{t_n}^t \sigma(\mathbf{r}(s)) ds \right). \quad (3-1)$$

$T(t)$ 表示沿光线从 t_n 到 t 的累积透射率, 即 t_n 到 t 光线没有碰到任何粒子的概率。从连续的神经辐射场中渲染一个视图需要为跟踪到期望虚拟相机的每一条相机光线估计积分 $C(\mathbf{r})$ 。

计算机是无法计算连续积分的, 因此需要使用数值估计的方法, 利用求面积法估计连续积分。确定性求面积通常用于渲染离散体素网格, 将有效地限制所表示的分辨率, 因为 MLP 仅在固定的离散位置上查询。相反, NeRF 使用分层抽样方法将 $[t_n, t_f]$ 放入 N 个均匀分布的容器中, 然后从每个容器中随机抽取一个样本:

$$t_i \sim U \left[t_n + \frac{i-1}{N} (t_f - t_n), t_n + \frac{i}{N} (t_f - t_n) \right]. \quad (3-2)$$

虽然使用的是离散的样本集去估计积分，但是分层采样使其能表示连续的场景，这是因为它使得 MLP 在优化过程中可对连续的位置进行评估。最终是通过这些样本来估计 $C(r)$ 。

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \text{ 其中 } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \quad (3-3)$$

在公式 3-3 中， $\delta_i = t_{i+1} - t_i$ 是相邻两个采样点的距离。上述函数对于计算 $\hat{C}(\mathbf{r})$ 的一组 (\mathbf{c}_i, σ_i) 值是可微的，使用 alpha 值 $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ 可以简化成传统的 alpha 合成。

3.2.3 神经辐射场的优化

上一节描述了将场景建模为神经辐射场的具体思路以及从该表示中渲染新视图所必需的核心组件。但是，NeRF 注意到这些组件不足以实现更高质量的神经渲染。NeRF 介绍了两项改进，以此表示高分辨率的复杂场景。第一种是输入坐标的位置编码，可帮助 MLP 表示高频函数，第二种是分层采样过程，可让有效地采样此高频表示。

3.2.3.1 位置编码

尽管理论上神经网络可以拟合任何函数，但是 NeRF 发现直接向网络 F_Θ 中输入 $xyz\theta\phi$ 会使得渲染效果不是非常好，在颜色和几何的高频变化上表现的非常差。这与 Rahaman 等人^[69] 最近的工作是一致的，该工作表明深度学习网络倾向于学习低频函数。他们还表明，在将输入传递到网络之前，使用高频函数将输入映射到更高维度的空间可以更好地拟合包含高频变化的数据。

在神经场景表示的背景下利用上述发现，把 F_Θ 重构成一个由两个函数组成的复合函数 $F_\Theta = F'_\Theta \circ \gamma$ ， F'_Θ 是学习得到的， γ 则是已知的，这能够明显地提升合成新视图的质量。 γ 建立了 \mathbb{R} 到更高维空间 \mathbb{R}^{2L} 的映射， F'_Θ 仍是一个常规的 MLP。正式使用的编码函数如下：

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)). \quad (3-4)$$

$\gamma(\cdot)$ 函数作用于 \mathbf{x} ($\mathbf{x} \in [-1, 1]^3$) 的每个坐标值和 \mathbf{x} 视角方向的单位向量 \mathbf{d} 的每一个坐标值。

3.2.3.2 分层体积采样

接下来介绍一下 NeRF 中最核心的部分，分层体积采样。一般的渲染策略是，在每条相机光线上密集地采样 N 个查询点，然后通过神经辐射场网络去评估得到

颜色和体密度，但是这一渲染策略效率十分低下：自由空间和对渲染图像无贡献的遮挡区域仍会被重复采样。NeRF 从体绘制中的早期工作^[70] 中汲取灵感，并提出一种分层表示，通过按比例分配样本到最终渲染的预期效果来提高渲染效率。

NeRF 同时优化两个神经网络：一个是 coarse 网络，另一个是 fine 网络。首先使用分层采样采集一组有 N_c 个位置的样本，在这些位置使用公式 3-2 和公式 3-3 来评估 coarse 网络。基于这个 coarse 网络的输出，就可以沿着每条光线对点进行更准确的采样，此时的样本点更偏向于物体附近。为此，首先将 3-3 中的 coarse 网络 $\hat{C}_c(\mathbf{r})$ 的 alpha 合成颜色重写为沿射线的所有采样颜色 c_i 的加权和：

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, w_i = T_i (1 - \exp(-\sigma_i \delta_i)). \quad (3-5)$$

将上述权重进行归一化 $\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_c} w_j}$ 可以生成一个沿着相机光线分段常数的概率密度函数。我们使用逆变换采样从该分布中采样第二组 N_f 个位置，最终使用公式 3-3 来计算最终沿着光线渲染的颜色，此时使用所有 $N_c + N_f$ 个采样点。

由于优化的是两个网络，因此整个 NeRF 的 loss 可以表示为：

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\|\hat{C}_c(\mathbf{r}) - C(\mathbf{r})\|_2^2 + \|\hat{C}_f(\mathbf{r}) - C(\mathbf{r})\|_2^2 \right]. \quad (3-6)$$

这个也是本文的研究动机 (motivation)，通过优化上述的采样过程，使 NeRF 可以减少训练一个网络，这是本文最核心的改进思想。

3.3 基于神经辐射场的新视图合成加速方法

针对 NeRF 采用分层抽样这一做法，即同时优化 coarse 网络和 fine 网络，通过 coarse 网络的输出去估计公式 3-5 中的权重 w_i 的分布情况，生成一个分段常数的概率密度函数，通过此分布进行二次采样并嵌入 fine 网络中，这在 NeRF 中证明是能显著提升渲染质量的。但也正是由于增加了一个网络，那么在网络前向传播这一部分的开销增加了一倍，这在实际应用中是难以接受的。为此，本文将此问题看作是一个领域知识的优化的问题。优化的目标在于无论是训练还是在测试过程中，只使用一个网络，这样理论上渲染一张新视图的时间会缩短一半。但是，在 NeRF 消融实验中证明了单纯地使用一个网络势必会降低新视图渲染的质量，因此必须采取一定的方式去补偿减少一个网络带来质量下降的问题。接下来首先会介绍本文提出的一个最基本的加速优化模型；然后详细介绍该模型中的核心组件查询表的构建，其中包含网络架构和如何建立对查询表的快速索引。

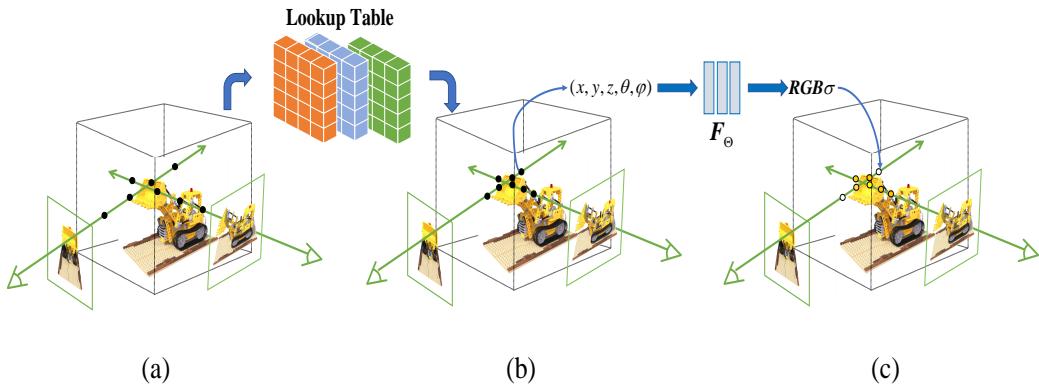


图 3-2 神经辐射场的新视图合成的加速框架。(a) 沿着相机光线采样 5D 坐标 (位置和观测方向); (b) 通过缓存的特征查询表找到采样点中距离表面最近的点，并在此点附近进行二次采样将; (c) 将 (b) 中位置 (x, y, z) 和方向 (θ, ϕ) 馈入 MLP 以生成颜色 RGB 和体密度 σ

3.3.1 模型综述

本小节将详细介绍本文提出模型的设计方法和细节。如图 3-2 所示，本文提出了基于神经辐射场的快速合成新视图的框架，为 NeRF 的推理过程进行加速。本文的框架一共包含三个步骤，第一步是网络裁剪，第二步是提取查询表，第三步是优化采样过程。

对于网络裁剪来说，由于 NeRF 是使用了 coarse 网络和 fine 网络来进行分层采样，为了操作方便，本文是去掉了 coarse 网络，仅保留 fine 网络，这显然会降低渲染的质量水平，因此还需要后面两个步骤的操作。对于缓存查询表的过程，这里先简单理解为建立了一个立方体网格，将格点坐标输入到 NeRF 预训练好的网络中，根据体密度为正值这一条件筛选出场景内部点的坐标，查询表真正存的是网络中间层（第四层的输出）的特征，查询表实质表征了物体的近似表面，具体的查询表的构建详见下一小节。对于第三步，我们通过缓存的查询表完成对采样点的优化，过程的概况为首先将粗采样的采样点（均匀采样完成的）馈入查询表中，若发现光线上所有点均不在查询表内则忽略该条光线，下面是有交集的情况，沿着光线的方向找到相机光线上第一个在查询表内的点，以此点为中心，适当的区间长度进行二次均匀采样，这些采样点即最终的采样点，位于物体表面附近，对渲染图像贡献较大。之后将高贡献的采样点馈入 NeRF 原有的网络框架（即 fine 网络）中输出颜色和体密度，通过积分计算出像素的 RGB 值，然后可以根据该 RGB 值与 ground truth 对应像素的 RGB 计算出 loss 并反向传播优化整个网络，最后我们就获得了一个能快速合成新视图的网络框架。

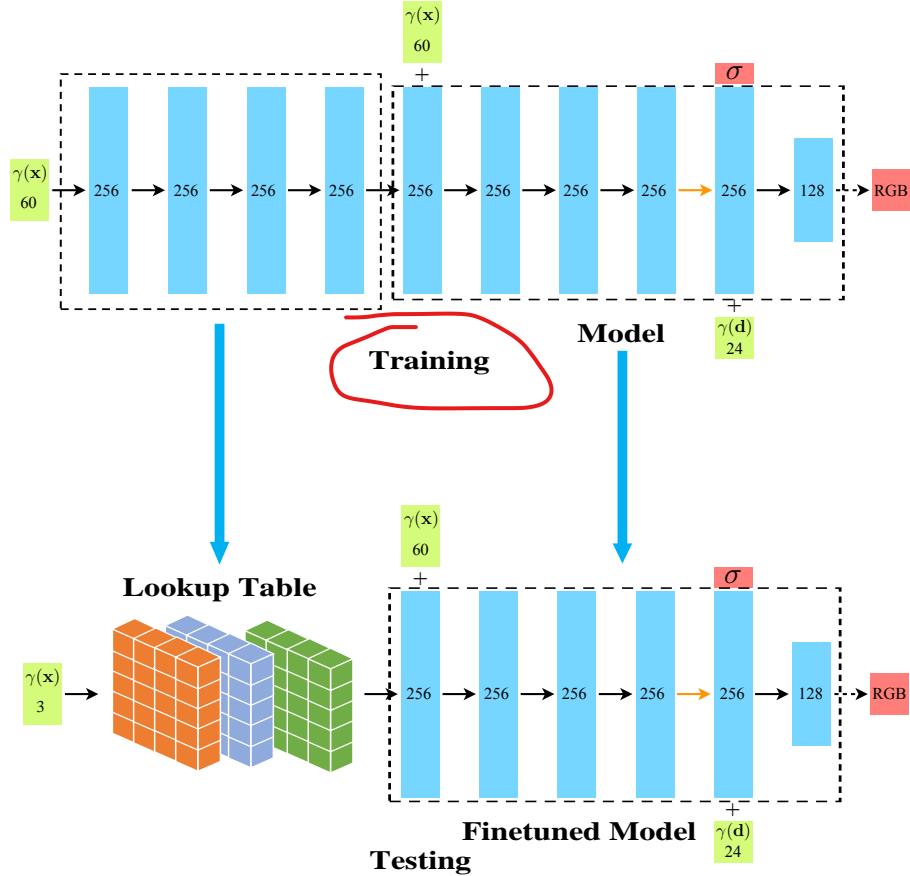


图 3-3 查询表的构建以及类似 justlookup^[68] 的架构，通过缓存网络特征来进行加速，并通过重新训练的方法对近似的特征进行学习。

3.3.2 查询表的构建

本节将详细介绍前一节中查询表的构建，主要介绍查询表的数据类型，生成方式，快速索引的方法，以及如何压缩 NeRF 的网络。

如图 3-3 所示，本文主要的构建方法是首先训练一个逐点网络，即使用 NeRF 的原网络进行预训练，将网络的前四层全连接层提取出来，我们可以得到一个逐点函数 \mathcal{G} ，类似 justlookup^[68]，我们使用查询表技术并利用数组索引的方法能够快速地获得近似的函数 $\hat{\mathcal{G}}$ 。

详细地，由前面的理论可知，我们可以用相机光线 $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}, t \in [t_n, t_f]$ 表示一组采样点的集合，为了使用方便，事先对光线的坐标范围进行尺度缩放，使得 $\forall t \in [t_n, t_f], \mathbf{r}(t) \in [-1, 1]^3$ 。

为表示方便，先忽略位置编码函数 $\gamma(\cdot)$ 的作用，假设逐点网络输入的是三维坐标点，那么可以将上述的映射 \mathcal{G} 表示为 $\mathcal{G} : \mathbb{R}^3 \Rightarrow \mathbb{R}^l$ ，其中 l 在是第四层全连接

网络输出变量的个数，在本文中 $l = 256$ 。更具体地， \mathcal{G} 是由四层 MLP 实现的，每层的输出的数目分别是 256, 256, 256, 256。

为了学习 \mathcal{G} ，我们利用原网络剩下的部分记为 Model。使用公式 3-6 中的均方误差联合优化这个端到端的网络，最终得到三元函数 \mathcal{G} 。一旦我们获得了 \mathcal{G} ，我们就可以通过构建一个查询表来获得近似的函数 $\hat{\mathcal{G}}$ 。

显然，我们现在的最重要的一步是为 \mathcal{G} 构建查询表。为了方便，我们还是假设输入被缩放到一个固定大小的立方体 \mathcal{V} 里，其中 $\mathcal{V} = [-1, 1]^3$ 。接着将 \mathcal{V} 分成规则的 M^3 个相等的体素块。那么每一个体素的边长则为 $\Delta = \frac{2}{M}$ 。

区别于 justlookup^[68]，我们并没有采取 $T[i][j][k] = \mathcal{G}(i\Delta, j\Delta, k\Delta), (i, j, k) \in [0, M]^3$ ，即存下所有格点所对应的特征，这样在 NeRF 这样细粒度的渲染工作中几乎是不可取的。实际中，我们需要使用 float32 的数据类型去存储，那么总内存开销就会达到 $32M^3 l bits$ ，以 $l = 256, M = 256$ 为例，则需要 16 GB 的内存，随着查询表的边长 M 增加，内存是以三次方的速度在增长，这在实际操作中是不可行的。

为此，我们必须采取相应的措施去压缩查询表。受 DeepSDF^[7] 的启发，在 DeepSDF 这篇工作中，可以通过输出的 SDF 值，利用 marching cubes 算法^[71] 重建物体的几何结构，类似地，我们知道 SDF 的物理含义是某一三维坐标点到物体表面的最短距离，负值表示在物体内部，正值表示在物体外部，零值则是物体表面；那么，NeRF 输出的体密度信息也有着相似的性质，体密度的物理含义是相机光线终止于某一点的概率，那么同样地，体密度为零值也可以表征物体的表面，图 3-4 就

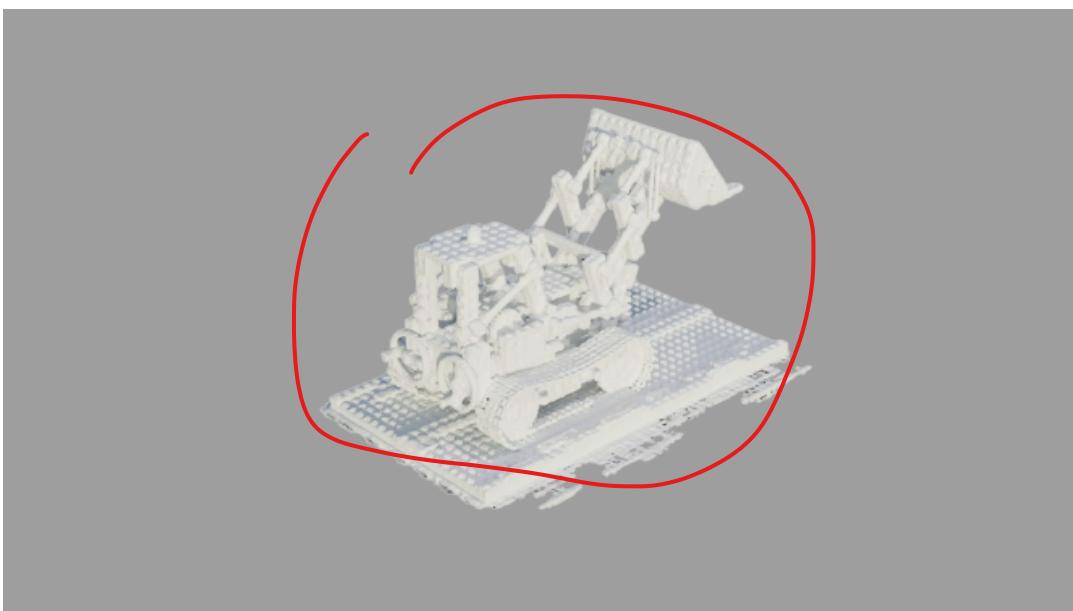


图 3-4 使用 NeRF 网络（体密度 > 0 ）提取出的 lego 的网格结构

是用 NeRF 模型提出 lego 的 mesh 结构。根据公式 3-5 可以看出当体密度为负值时权重 w_i 是负值是没有意义的，这也与 NeRF 的处理方式相同，NeRF 在体密度输出的时候使用了 ReLU 函数，因此我们可以认为负值的体密度对渲染的贡献可以忽略。那么结合上一节，如果我们可以将采样点约束在物体表面附近，那么将能获得更好的渲染效果，这可以补偿减少一个网络带来的质量下降。

基于以上的分析，我们假设正值的体密度表征物体的内部，接下来就可以对原查询表进行压缩。我们的目标是只存 \mathcal{V} 中输出的体密度大于零的格点，这样可以大大减少内存的开销。

具体地，我们记从三维坐标点到输出体密度这一部分网络对应的函数为 Q ，此时压缩后的查询表为：

$$T_{shrinked} = \{G(i\Delta, j\Delta, k\Delta) | Q(i\Delta, j\Delta, k\Delta) > 0, (i, j, k) \in [0, M]^3\}. \quad (3-7)$$

至此，我们获得了最终的查询表，由于此查询表是不规则的，或者可以认为是稀疏矩阵，因此索引此查询表也跟常规的方法不同，接下来将详细介绍经过压缩后的查询表的索引。

对于 $T_{shrinked}$ 的存的每一个格点 $(i\Delta, j\Delta, k\Delta)$ ，我们必须将每一个常规索引 (i, j, k) encode 到一个唯一值 code，换句话说，我们的目标是使得 \mathcal{V} 中的任何一个格点对应的 code 都是不重复的。为此，我们可以使用以下的映射方式：

$$\mathcal{E}(i, j, k) = i + j \cdot M + k \cdot M \cdot M, (i, j, k) \in [0, M]^3, \quad (3-8)$$

这个对索引的编码操作可以使得将 \mathcal{V} 中的所有格点都区分开。

接下来，我们将建立一个 code 到 $T_{shrinked}$ 的查询表 \mathcal{H} 。具体的建立方式为，

- a) 首先建一个大小 $size = M + M^2 + M^3$ 的查询表 \mathcal{H} ，使之能装下 \mathcal{V} 中所有格点；
- b) 按照 $T_{shrinked}$ 对应的所有点的顺序，假设其中一个点对应的常规索引为 (i, j, k) ，该点在 $T_{shrinked}$ 的位置为 index 对其使用 encode 操作得到 $code = \mathcal{E}(i, j, k)$ ；
- c) 之后填充查询表 \mathcal{H} ，具体做法是 $\mathcal{H}[code] = index$ 。

严谨来说，非 $T_{shrinked}$ 的点在索引时，其 code 可能会超过 $T_{shrinked}$ 的索引范围，因此，我们必须处理此问题。具体做法是，在 $T_{shrinked}$ 的最后面加一个哨兵，即增加一个 l 维的全零的向量，假设目前 $T_{shrinked}$ 存了 S 个点对应的特征。而 \mathcal{H} 在初始化的时候，初始值都设置为 $S - 1$ ，即当 (i, j, k) 不在查询范围内时，其 code 可以通过 \mathcal{H} 映射到 $S - 1$ ，那么将获取 $T_{shrinked}[S - 1]$ 的值，从而得到一个鲁棒的索引过程。

至此，我们可以总结整个索引的过程。给定一个空间直角坐标系的三维坐标点 (x, y, z) ，首先通过下面式子计算其常规索引：

$$(i, j, k) = \left(\left\lfloor \frac{x+1}{\Delta} \right\rfloor, \left\lfloor \frac{y+1}{\Delta} \right\rfloor, \left\lfloor \frac{z+1}{\Delta} \right\rfloor \right), \quad (3-9)$$

有了常规索引后，需要对其进行 `encode`，得到 $code = \mathcal{E}(i, j, k)$ ，之后通过 \mathcal{H} 查询到 $code$ 对应的在 $T_{shrinked}$ 中的索引，则整个查询过程为：

$$T[i][j][k] = T_{shrinked}[\mathcal{H}[\mathcal{E}(i, j, k)]]. \quad (3-10)$$

最终，如果我们提前缓存了 T ，则可以使用 $T[i][j][k]$ 的值去估计 $\hat{\mathcal{G}}(x, y, z)$ ，因此 $\hat{\mathcal{G}}$ 的将是非常快的，它的开销只取决于访问内存的时间。

显然，我们直接将 $\hat{\mathcal{G}}$ 镶入到经过训练的模型 *Model* 中会使渲染效果下降的，因为 $\hat{\mathcal{G}}$ 并不完全正确等于 \mathcal{G} ， $\hat{\mathcal{G}}$ 仅仅是 \mathcal{G} 的一个近似。因此为了保持渲染质量不下降，我们在构建了查询表 T 后，需要基于 $\hat{\mathcal{G}}$ 对 *Model* 进行再训练（或者叫微调），这一操作能显著提升渲染质量，在 $M = 200$ 的时候甚至可以达到和原始 NeRF 质量相当的性能。图 3-3 表征了 \mathcal{G} 是如何由我们的方法实现的。

3.3.3 查询表与本文方法的结合

我们在上一小节详尽地叙述了查询表的创建过程，包含查询表的构建，索引，压缩，还有通过近似 \mathcal{G} 来进行加速，通过再学习将损失的质量调回原 NeRF 水平。接下来，本节将详细介绍查询表的第二重加速功能：将查询表应用于 NeRF 的采样过程。

如果只是使用查询表去近似 \mathcal{G} ，那么这并没有完全展现出本文最核心的方法。实际上，在上一小节中，我们不止做了这么多工作。我们都知道，在查询表的构建过程中，本文是对查询表进行了压缩的操作，这表面上只是为了减少查询表内存开销，事实上我们还应用了 NeRF 体密度的特性。

由于我们是使用的体密度 $\sigma > 0$ 这一阈值条件对 \mathcal{V} 中格点进行筛选，仅在 $T_{shrinked}$ 中保留其体密度为正值的点所对应的特征（即 l 维向量），并且我们还在 $T_{shrinked}$ 的最后放置了哨兵来表征没有查询到相应点的特征。

因此，我们可以合理地使用 \mathcal{H} 和 $T_{shrinked}$ 这两个查询表对输入的点进行判断，显然如果可查询到，则该点在物体内部，否则不在物体内部。具体地，由于设置了哨兵，我们实际可以只使用索引转换查询表 \mathcal{H} 对内外进行判断。对于任意的常规索引 $(i, j, k) \in [-1, 1]^3$ ，计算 $\mathcal{H}[\mathcal{E}(i, j, k)]$ 的查询结果，若结果为 $S - 1$ ，则原世界坐标 (x, y, z) 则不在物体（场景）的内部，若结果不为 $S - 1$ ，则在物体内部。

有了以上的理论基础，接下来将详尽阐述是如何利用查询表进行采样优化以

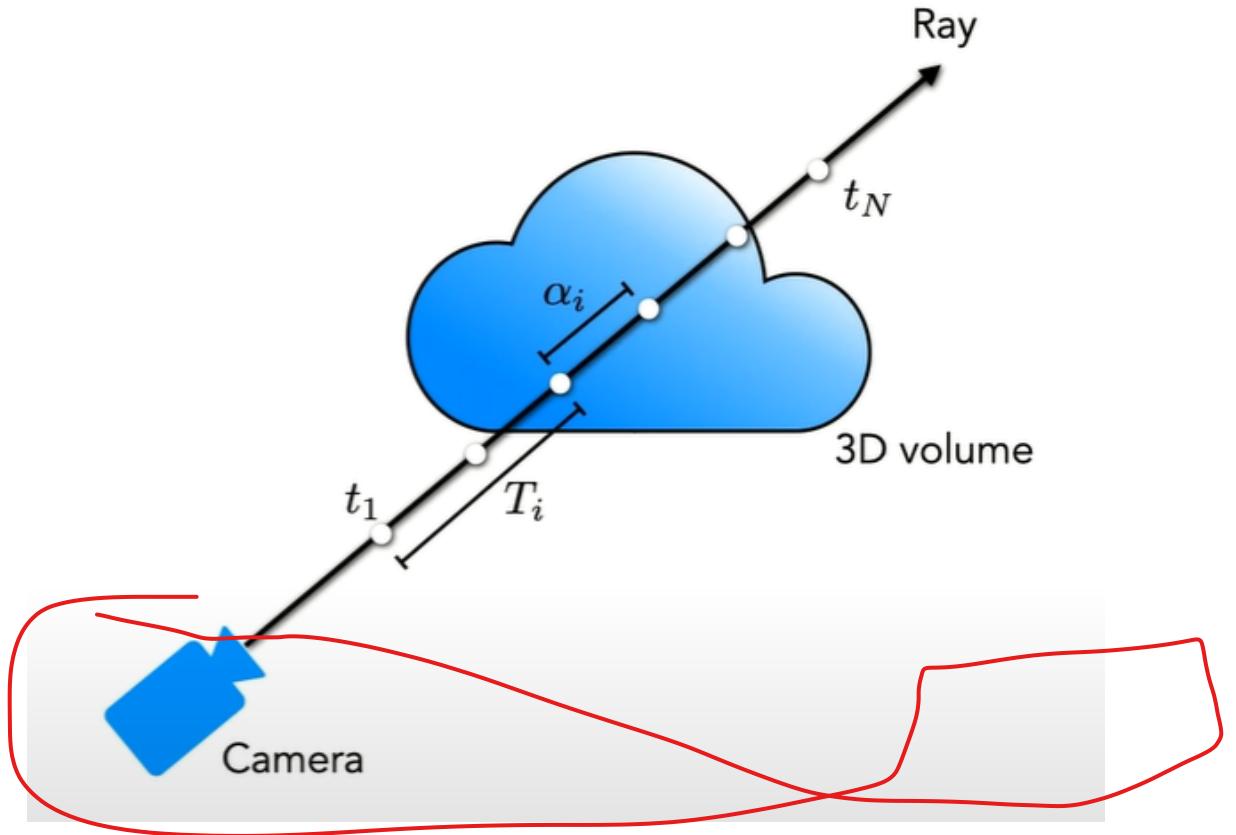


图 3-5 体绘制的采样示意图

及加速。

针对前面两节的概述，我们知道，NeRF 是采用分层抽样的方法，即同时优化 coarse 网络和 fine 网络，通过 coarse 网络的输出去估计公式 3-5 中的权重 w_i 的分布（概率密度函数），这是通过增加网络来提高渲染质量的方法。但同时也很显然，使用两个网络，那么时间开销会增加一倍。

为了加速 NeRF，我们的目标是去掉一个网络，仅使用一个网络来进行优化和渲染，但是这势必会使得渲染质量下降。考虑到 NeRF 使用两个网络是为了将采样点位置优化到物体周围，这一点启发了我们，如果使用上述的查询表将采样点约束到物体表面附近，那么渲染质量会比仅使用一个网络显著提升，通过实验证明本文的方法可以在几乎不损失渲染质量的情况下对 NeRF 的推理过程进行加速。下面是查询表是如何和本文方法进行融合的。

如图 3-5 所示的是体绘制的采样示意图。假设相机光线为 $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ ，并在 $[t_n, t_f]$ 的范围内均匀采样 N 个点，深度记为 t_1, t_2, \dots, t_N ，这些深度是递增的，对应的点的世界坐标为 $\mathbf{r}(t_1), \mathbf{r}(t_2), \dots, \mathbf{r}(t_N)$ 。分别计算这些点的常规索引，以及编码之后的 code，通过这些 code 去查询 \mathcal{H} ，得到一个长度为 N 的向量 $vector$ 。之后按顺

序对 *vector* 的每一个坐标值进行判断, 若值均为 $S - 1$, 则说明该光线与物体没有交点, 则采样点不需要优化, 若存在值不为 $S - 1$ 的, 则说明光线和物体有交点, 记第一个交点对应的深度值为 t_{mid} , $mid \in \{1, 2, \dots, N\}$ 。在有交点的情况下, 我们需要重新设定采样区间长度为 L , 则新的采样区间为 $[t_{mid} - L/2, t_{mid} + L/2]$, 此区间能表征物体的表面附近, 我们需要在此区间内重新均匀采样 N 个点, t'_1, t'_2, \dots, t'_N , 然后就可以将 $\mathbf{r}(t'_1), \mathbf{r}(t'_2), \dots, \mathbf{r}(t'_N)$ 这些采样点送入网络进行训练, 最终实验证明使用了改进的采样方式, 在我们减少一个网络的时候, 可以保证渲染质量几乎不下降。

最终, 我们通过缓存一个查询表, 成功地在不损失精度的情况下减少了一个网络的开销, 对渲染过程进行了加速, 那么因此, 本文方法对应的 loss 函数变为:

$$\mathcal{L} = \sum_{\mathbf{r} \in R} \left[\|\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\|_2^2 \right]. \quad (3-11)$$

3.4 本章小结

本章首先介绍了本文所研究的问题定义。接着介绍了新视图合成任务中的神经辐射场表示方法, 然后在此基础上详细地介绍了本文基于问题而提出的方法论, 包含模型网络结构, 查询表技术的引入与查询表的构建, 以及如何把查询表应用到本文的方法中。

综上, 本文借助查询表技术替换了一部分网络(逐点网络)参数, 使得查询可以直接从内存(或缓存)中获取中间层的特征, 这可以显著加速渲染过程; 为了减少查询表的内存开销, 同时也是为了与本文方法进行融合, 我们对查询表进行了压缩的操作, 具体是使用体密度 $\sigma > 0$ 这一阈值条件去筛选位于物体内部的格点; 本文的核心方法是减少 NeRF 的原始的两个网络的结构, 我们仅采用一个网络进行训练和渲染, 为了保证质量不下降, 使用查询表判断原始采样点并第一个在物体内的点, 并在此附近进行重新的均匀采样, 可以补偿只用一个网络带来的质量下降; 当然, 最关键的一点是, 将上述方法结合在一起时, 由于查询表替换网络这一部分是用的近似特征, 因此为了保证质量不下降, 必须进行近似特征再学习。最终, 通过本文的方法可以实现在几乎不损失精度的情形下对 NeRF 合成新视图进行加速。

第四章 实验部分

为了验证本文提出的加速 NeRF 的方法的有效性，本章将详尽地描述所做的相关的验证性和解释性实验，首先将本文提出的方法命名为 F-NeRF。首先将介绍实验的基本设置，包含使用的深度学习框架、开发语言、训练集和测试集，其次给出 F-NeRF 在合成新视图这样任务上的基本效果，然后详尽地描述 F-NeRF 的预训练过程，训练过程，相关参数配置，接着描述 F-NeRF 在合成数据集 Synthetic-NeRF 和真实场景数据集 LLFF-NeRF 上的实验效果以及分析。最后根据实验效果给出实验结论并进行总结。

4.1 实验基本设置

首先先介绍下本文实验的基本设置，包含开发环境和实验所需的数据集。

4.1.1 开发环境

现如今的深度学习框架有很多，包含 Caffe、Tensorflow 和 Pytorch 等。Caffe 是基于 C++ 语言开发出的深度学习框架，当然也提供 python 的接口，核心是使用 C++ 开发的，也因此更高效，适合工业界开发，但它缺少灵活性和拓展性，环境配置复杂。Tensorflow 是目前应用最广泛的深度学习框架，由谷歌公司开发与维护，具有十分成熟的社区与工具，已经广泛地被应用到企业中了。Pytorch 是相对较新的深度学习框架，由 Facebook 开发与维护的，以代码简洁上手容易，容易实现并行等优点进入到开发者的视野中，现在越来越受到学术界和工业界的青睐。由于查询表使用 Pytorch 实现更为简便，所以本文使用 Pytorch 来实现神经辐射场的所有结构。使用其他框架也可以完全复现本文的方法。本文基于 Ubuntu 18.04 操作系统进行开发，开发语言为 python 3.6。使用的 GPU 是一张显存为 12GB 的 NVIDIA Titan X。

4.1.2 数据集

在整个新视图合成的实验中，本文使用的都是广泛使用的公开的数据集，包含合成数据集 Synthetic-NeRF 和真实世界数据集 LLFF-NeRF。下面将大致介绍下这两种数据集。

1) Synthetic-NeRF 数据集：这个合成数据集是 NeRF 生成并使用的，包含由

blender 渲染的八个物体，每一张图都是 800×800 的分辨率。本文为了方便测试，实验中实际使用的是分辨率 400×400 的图像。对于每一个物体，有 100 个视角作为训练集，200 个新视角作为测试集。Synthetic 数据集的优点是，它拥有绝对准的相机位姿，这能够排除受位姿影响而带来的质量的下降。如图 4-1 是该数据集的示意图。

- 2) LLFF-NeRF 数据集：这个真实数据集包含八个前向场景的图像，其中五个来自于 LLFF^[54]，三个来自于 NeRF。每个场景包含 20 到 62 张图，其中 1/8 用作测试。所有的图像都有着 1008×756 的高分辨率。本文为了方便测试，实验中实际使用的是分辨率 504×378 的图像。如图 4-2 是该数据集的示意图。



图 4-1 Synthetic-NeRF 数据集示意图，第一行是部分训练集，第二行是部分测试集



图 4-2 LLFF-NeRF 数据集示意图，第一行是部分训练集，第二行是部分测试集

4.2 实现细节

本文提出的 F-NeRF 架构具体见 3.3 节。本文的加速模块是缓存了一张查询表，查询表的位置信息反映了物体的近似几何结构，具体缓存的是第四层全连接层输出的特征。

为了得到缓存所需的查询表，首先需要基于 NeRF 网络进行预训练，具体参数为，每个 batch 使用 1024 条光线，相应地，每条光线上的采样点数目为 $N_c = 64$ (*coarse* 网络)， $N_f = 128$ (*fine* 网络)。同样地，使用 Adam 优化器^[72]，在学习的过程中，学习率从 5×10^{-4} 指数衰减到 5×10^{-5} 。(其他的 Adam 超参数使用的是默认值， $\beta_1 = 0.9$ ， $\beta_2 = 0.999$ ， $\epsilon = 10^{-7}$) 在合成数据集上，对每个物体的训练过程在本文的设备上大概要训练 100000 epoch (大约 12 小时)。真实数据集上，对每个场景的训练大概 200000 到 300000 epoch 会收敛 (1 到 2 天)。

得到了预训练的 NeRF 模型后，接下来就可以训练本文的模型 F-NeRF。首先要通过 NeRF 网络缓存一张特征查询表 T ，这个过程大概需要半分钟。为了保证质量几乎不下降，由于本文提出的 F-NeRF 模型只使用了一个网络，那么再使用 *coarse* 网络的 64 个采样点就不足以从查询表中探测到物体表面，因此，为了权衡速度与质量，F-NeRF 使用了 $N = 128$ 个采样点，这显然仍然少于 $N_c + N_f = 192$ 。在另一个加速的部分，F-NeRF 使用缓存的 T 替换网络的前四层全连接层，这样在测试的过程中就可以直接从缓存中获取特征，这大大减少了时间开销。以上加速模块的引入显然会导致质量的下降，有两点，第一点，去掉了 *coarse* 网络，减少了采样点，第二点，使用了缓存的特征查询表的近似特征，因此，为了保证质量不下降，必须对 F-NeRF 网络进行再训练，两个数据集大约训练 100000 epoch 网络可以收敛，将近 4 小时。

在测试时，本文的方法和 NeRF 一样都需要一组相机参数作为输入，作用于为输出图像的每个像素发出的光线。沿着每条相机光线生成的大量采样点按照 NeRF 的理论进行积分。然而 F-NeRF 能够使用缓存的特征查询表对单网络的采样点进行优化。为了提升性能，本文对将第一次的均匀采样点送入到缓存的查询表中进行查询 (使用的是最近邻的插值方式)，由于查询表表征的是物体的几何结构表面，因此沿着光线找到第一个击中物体表面的点 (或缓存命中)，以此为中心在适当范围内进行二次均匀采样，之后将新采样点送入网络，第一步加速完成。为了进一步提升效率，F-NeRF 借助了 justlookup^[68] 的思想，查询表缓存的是原 NeRF 网络前四层输出的特征，采样点可以直接通过最近邻内插法在查询表内查询相应的特征，而减少了网络的正向传播层数，这大大加快了渲染的速度。本文的未压缩前的查询表对应的分辨率大小为 200^3 ，这是实验硬件设备可支持的最大分辨率。 200^3

分辨率的查询表实质上对应着 7.63 GB 的内存开销，但是经过本文方法进行压缩后，平均仅需要 0.97 GB，有效地去除空白的自由空间，这极大地节省了内存，同时也使得查询表隐藏着场景的几何信息。

4.3 评测指标

4.3.1 图像质量指标

为了评估合成的新视图的质量，实质都是计算合成图像与真实观测的图像之间的差异度。假定合成图像为 \mathbf{Y} ，真实图像为 \mathbf{X} ，图像均为 $M \times N$ 的彩色三通道图像，假设每个像素使用八位来存储，即像素值范围为 [0, 255]。本文使用的质量评估指标如下：

1) 峰值信噪比 (Peak Signal to Noise Ratio, PSNR)

PSNR 是一个信号处理学科上的概念，物理含义是信号可能的最大功率与噪声功率的比值，一般使用对数分贝来表示。对于图像的情形，可以有以下公式：

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}^2} \right),$$

其中均方误差 $\text{MSE} = \frac{1}{3MN} \sum_{R,G,B} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [\mathbf{X}(i,j) - \mathbf{Y}(i,j)]^2,$ (4-1)

PSNR 是相对客观的指标，他是逐像素计算的。由公式 4-1 可以看出，PSNR 值越大，对应 MSE 越小，那么合成图像和真实图像直接的差异也就越小。所以，PSNR 这一指标在一定程度上可以反映图像质量的好坏，但是它是比较客观的，不太涉及人眼观感，因此 PSNR 值越大，在人眼看来可能不一定好。所以在对比图像质量的时候，仅仅使用 PSNR 是不够的。

2) 结构相似性度量 (Structural Similarity Index Measure, SSIM^[73])

结构相似性的基本理念是认为图像是高度结构化的，图像之间的相邻像素有着很强的关联性。人类的视觉系统已经习惯了这种很强的结构化信息，因此在衡量图像质量的时候，推荐使用 SSIM 这一度量指标。具体地：

$$\text{SSIM}(\mathbf{X}, \mathbf{Y}) = [l(\mathbf{X}, \mathbf{Y})]^\alpha \cdot [c(\mathbf{X}, \mathbf{Y})]^\beta \cdot [s(\mathbf{X}, \mathbf{Y})]^\gamma, \quad (4-2)$$

其中 $l(\mathbf{X}, \mathbf{Y})$, $c(\mathbf{X}, \mathbf{Y})$ 和 $s(\mathbf{X}, \mathbf{Y})$ 定义如下：

$$l(\mathbf{X}, \mathbf{Y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \quad (4-3)$$

$$c(\mathbf{X}, \mathbf{Y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \quad (4-4)$$

$$s(\mathbf{X}, \mathbf{Y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \quad (4-5)$$

这里的 $l(\mathbf{X}, \mathbf{Y})$ 、 $c(\mathbf{X}, \mathbf{Y})$ 和 $s(\mathbf{X}, \mathbf{Y})$ 分别是亮度、对比度、结构。此外， μ_x, μ_y 是 \mathbf{X} 和 \mathbf{Y} 的均值， σ_x, σ_y 分别是 \mathbf{X} 和 \mathbf{Y} 的方差， σ_{xy} 是 \mathbf{X}, \mathbf{Y} 的协方差。 $C_1 = (k_1 \times 255)^2$ ， $C_2 = (k_2 \times 255)^2$ ， $C_3 = \frac{C_2}{2}$ 。其中， k_1, k_2 一般取 0.01 和 0.03。实验中， α ， β 和 γ 均置为 1。SSIM 的值越大，表明合成图像与真实图像结构越相似。

3) 感知相似度 (Learned Perceptual Image Patch Similarity, LPIPS^[74])

该指标是基于深度学习的度量方法，这是借助了深度学习能够提取到更细粒度的图像特征，并进行对比，该指标比 PSNR、SSIM 更能符合人类的感知情况。感知损失的计算方式为：

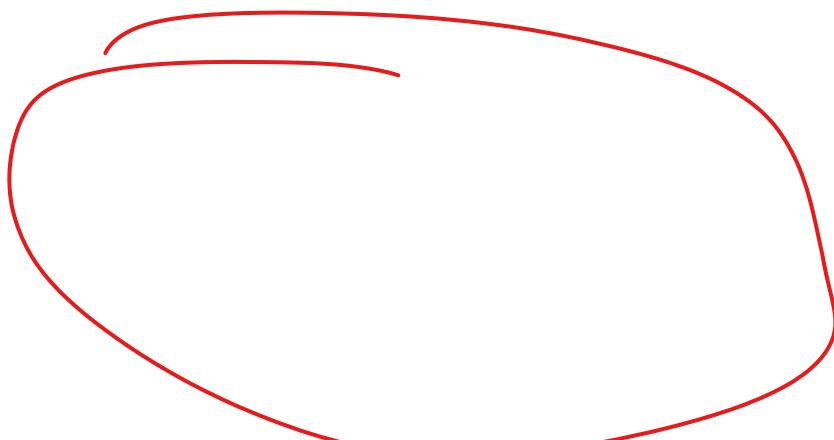
$$d(\mathbf{x}, \mathbf{x}_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|\mathbf{w}_l \odot (\hat{\mathbf{y}}_{hw}^l - \hat{\mathbf{y}}_{0hw}^l)\|_2^2, \quad (4-6)$$

这里 \mathbf{x}, \mathbf{x}_0 是输入的图像， l 是网络的层数， $\hat{\mathbf{y}}_{hw}^l, \hat{\mathbf{y}}_{0hw}^l$ 表示第一层网络的特征图， \mathbf{w}_l 为权重。上述也表明了 LPIPS 实际是计算了两张图之间的特征距离，因此值越小表征两张图像越相似。

结合上述三个质量指标，分别可以从客观，结构，人类感知多个维度来评价合成图像与原图的差别，这样的评价体系更完整、更准确、也更公平。

4.3.2 渲染速度指标

速度指标这里就比较简单，没有质量那么难以评估，因为速度是客观的，常规地，可以使用时间或者帧率对速度进行表征。



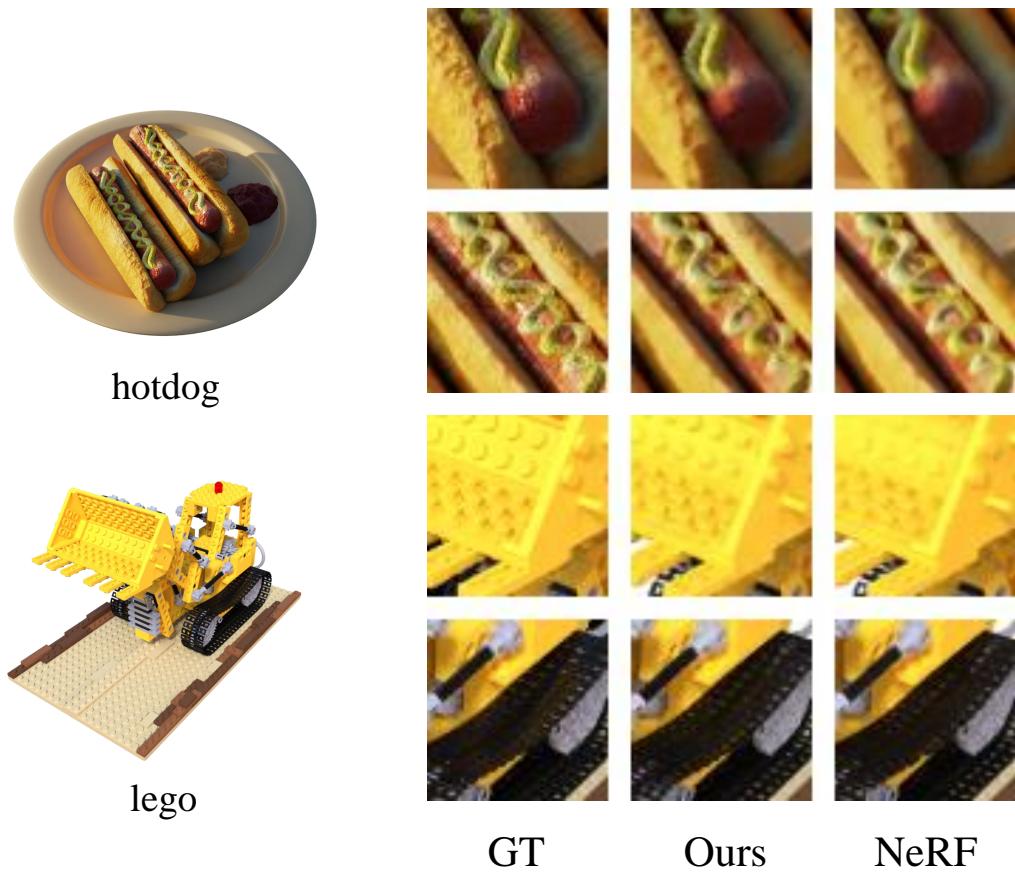


图 4-3 F-NeRF 与 NeRF 在合成数据集 Synthetic-NeRF 上的定性比较，使用的是分辨率 400^2 的图像。

4.4 实验结果与分析

下面介绍本文的实验结果与分析部分。本文分别在公开的合成数据集 Synthetic-NeRF 和真实数据集 LLFF-NeRF 上进行了新视图合成任务的实验，上述数据集的具体细节在 4.1.2 小节。分别在这两个数据集下对实验结果进行分析。

4.4.1 基于 Synthetic-NeRF 数据集的实验

为了验证本文提出的加速方法的有效性，首先本文在公开的合成数据集 Synthetic-NeRF 的八个物体上进行了实验验证。合成数据集的场景相对简单，位姿是绝对准确的，这对于评估质量来说很有帮助。图 4-3 给出了本文提出的 F-NeRF 与 NeRF 的定性对比结果。从结果可以看出，F-NeRF 几乎达到了和 NeRF 一样的质量，甚至在一些细节上，本文提出的 F-NeRF 还做的比较好，比如 hotdog 的沙拉酱部分，还有 lego 的铲斗部分，这一定程度上反映了本文提出的方法能够获取比较靠近表面的采样点，因为这些采样点对渲染过程的贡献更大。

表 4-1 本文提出的 F-NeRF 与 NeRF 在 PSNR, SSIM, LPIPS 以及渲染一帧图像的时间上的定量对比, 其中 PSNR 的单位是分贝 (dB), 时间的单位是秒/帧 (s/frame)

	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
PSNR ↑									
NeRF	31.21	24.72	28.19	33.03	31.01	28.65	30.08	26.43	29.16
Ours	32.11	24.42	27.84	35.10	31.15	28.79	31.53	26.14	29.64
SSIM ↑									
NeRF	0.966	0.921	0.962	0.958	0.950	0.941	0.972	0.835	0.938
Ours	0.968	0.908	0.955	0.972	0.958	0.951	0.985	0.817	0.939
LPIPS ↓									
NeRF	0.046	0.091	0.044	0.121	0.056	0.063	0.028	0.173	0.077
Ours	0.045	0.100	0.058	0.044	0.051	0.053	0.019	0.196	0.071
Time ↓									
NeRF	15.23	15.25	14.89	16.15	15.68	15.87	15.01	15.29	15.42
Ours	4.71	4.54	4.80	4.94	4.84	4.88	4.67	4.57	4.74

表 4-1 提供了 F-NeRF 与 NeRF 在合成数据集下的具体定量指标对比。从表格中可以看出, 本文提出的方法 F-NeRF 在几乎不损失精度的情况下能够在渲染速度上为 NeRF 加速 3.2 倍。质量方面, F-NeRF 在 PSNR 和 SSIM 这两项指标的均值略高于 NeRF, 在 LPIPS 这一指标上比 NeRF 低 0.006。

以上结果表明, 本文提出的新视图合成加速方法在合成数据集上能够为 NeRF 进行几乎无质量损失的显著加速。

表 4-2 真实世界数据集下的质量与渲染速度对比 (八个场景的均值)

	PSNR (dB) ↑	SSIM↑	LPIPS↓	Speed (s/frame) ↓
SRN ^[41]	22.84	0.668	0.378	-
LLFF ^[54]	24.13	0.798	0.212	-
NeRF ^[4]	26.50	0.811	0.250	14.33
Ours	26.87	0.836	0.226	4.23

4.4.2 基于 LLFF-NeRF 数据集的实验

真实场景相对于合成场景更为复杂，受光照条件、相机标定等因素的影响，因此合成高质量的新视角下的图像是相对困难的。本小节将本文提出的 F-NeRF 方法应用在 LLFF-NeRF 数据集上，与目前最先进的方法 SRN^[41]，LLFF^[54]，以及 NeRF^[4] 进行对比。真实场景的相机位姿都是通过 COLMAP^[6] 方法计算的。

图 4-4 给出了本文与 NeRF 在真实场景下的定性对比图，本文将图像中相对细节较多较复杂的部分使用红蓝框给框起来，读者可以自行放大进行对比。从图中可以看出，本文的方法在渲染的新视图的质量方面达到了几乎和 NeRF 相同的水平。具体地，对于图中的 T-Rex 场景，对应于红色框的部分，可以看出，本文的 F-NeRF 和 NeRF 都出现了细节的严重丢失，这是因为该物体过于细小，比较难采样到稳定的采样点，但事实上，F-NeRF 丢失的细节较 NeRF 还是要少一些，这是因为 F-NeRF 在采样上进行了优化，利用查询表缓存相关的几何信息，使得采样到的点是位于物体表面的对渲染贡献较高的部分。

最后表 4-2 给出了本文与其他方法的定量指标的对比（取八个场景的均值），可以看出本文提出的 F-NeRF 在 PSNR 上平均比 NeRF 提高 0.37 dB，在 SSIM 上能提升 0.033，在 LPIPS 上能降低 0.024，此外更能够给 NeRF 在渲染速度上加速 3.4 倍，这将使得基于神经辐射场的新视图合成系统的构建提供了可能，在下一章会具体详述本文基于 F-NeRF 所设计并实现的新视图快速合成的系统。

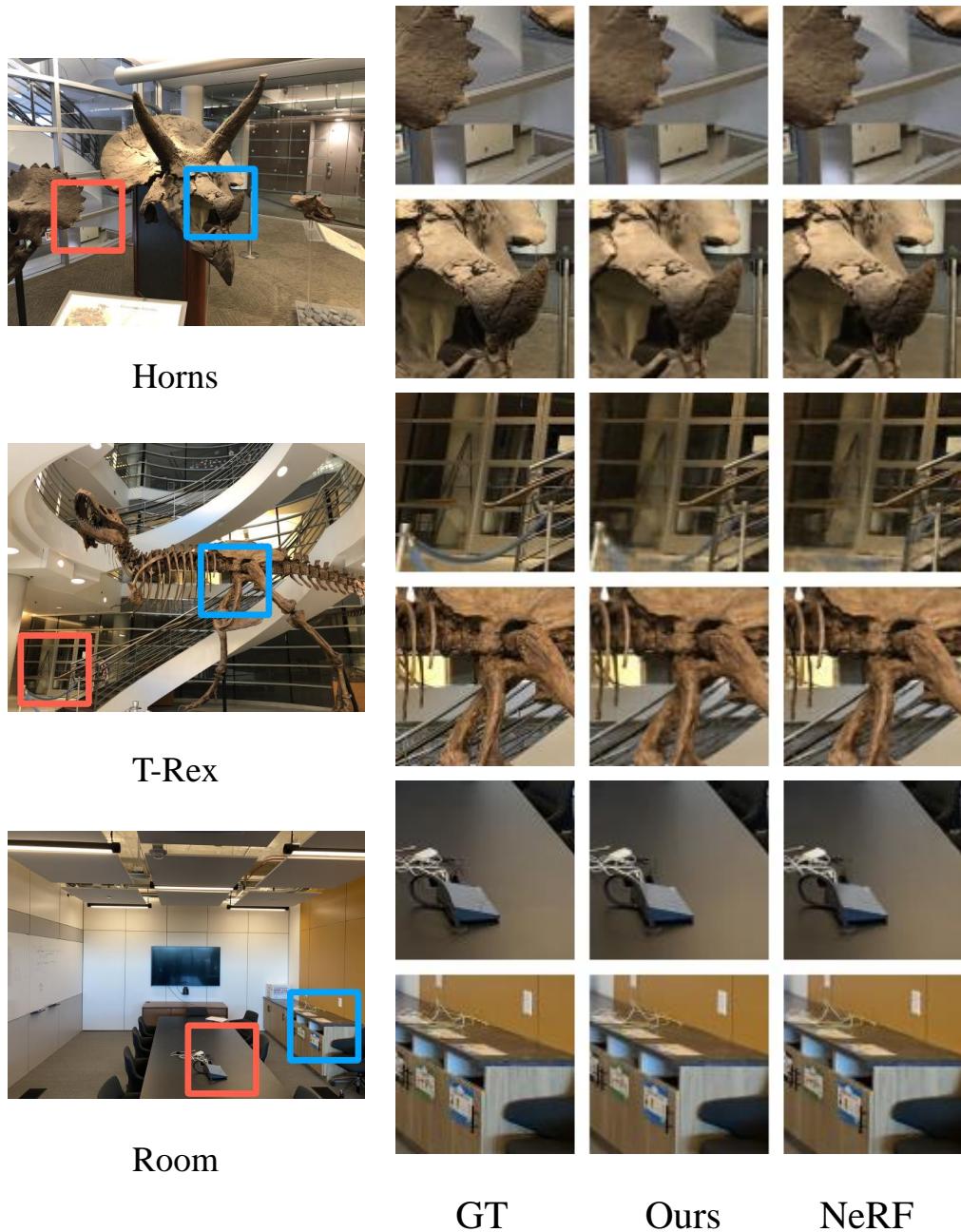


图 4-4 F-NeRF 与 NeRF 在真实场景数据集 LLFF-NeRF 上的定性比较，使用的是分辨率
为 504×378 的图像。

4.5 本章小结

本章主要介绍了本文的实验部分。本章第 4.1 节首先介绍了本文实验的开发环境、数据集。第 4.2 详细叙述了本文实验的一些细节，包括参数的设置，完整的训练过程，以及完整的测试过程。第 4.3 主要介绍了实验的具体评测指标，包含新视图合成任务中较为权威的质量评测指标，比如 PSNR、SSIM 和 LPIPS，这些指标涵盖了客观、结构相似性，以及人眼感知等多个层面去评估图像的质量，此外本节还提供了速度评估指标，主要是测试时间或渲染的帧率。最后分别基于公开的合成数据集 Synthetic-NeRF 和真实场景数据集 LLFF-NeRF，将本文提出的方法与 NeRF 进行了性能对比，从结果上面看，本文提出的方法几乎在不损失质量的情况下对 NeRF 进行明显加速。

第五章 系统设计与实现

第三、四章分别对本文提出的“基于神经辐射场的快速新视图合成方法”的方法论的详述和在公开数据集上大量的实验对比验证。为了将上述研究方法转成实际的系统，本文设计并实现了基于神经辐射场的快速新视图合成系统。此系统可以接入移动端三维传感器采集的多视角的图像，再通过本文提出的方法训练出高保真的神经辐射场，最终可以快速渲染出任意新视角的视图，可以和已知位姿的 ground truth 进行对比。

5.1 系统需求分析

5.1.1 功能性需求分析

基于上一节提出的系统设计的思路，本文基于神经辐射场的新视图快速合成系统可分为两大模块，分别是数据处理模块和算法模块，具体见图 5-1。

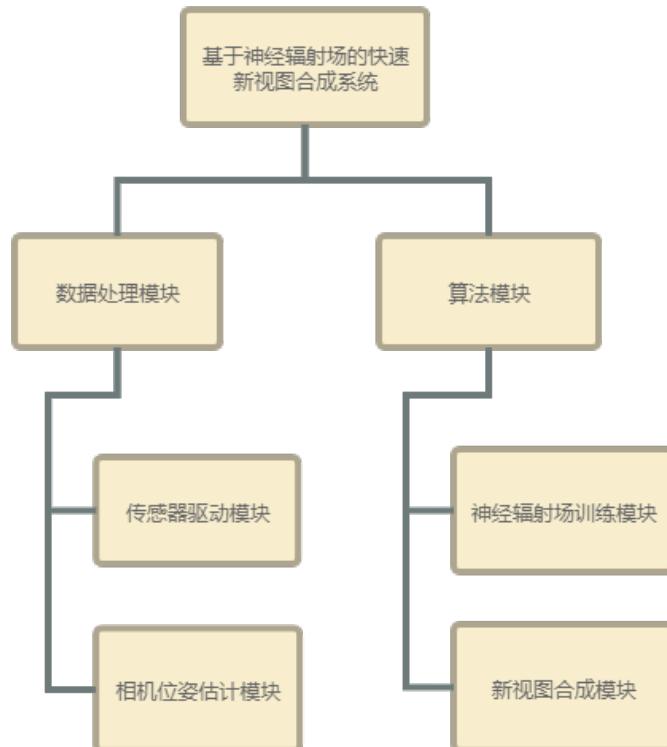


图 5-1 系统架构图

- a) 数据处理模块：包括以下两个模块内容：
 - 1) 传感器驱动模块：用于获取传感器的数据，例如 RGB 图像。
 - 2) 相机位姿估计模块：使用 COLMAP^[6] 实现对真实图像位姿的计算。
- b) 算法模块：包括以下两个模块内容：
 - 1) 神经辐射场训练模块：用于将已知位姿的图像训练成一个 5D 神经辐射场。
 - 2) 新视图合成模块：通过训练好的模型，给定位姿，合成该位姿下的新视图。

5.1.2 其他需求分析

在经典的软件质量模型“FURPS+”中，除了上一小节的功能性需求，还剩下四个非功能性需求，分别是易用性(Usability)、可靠性(Reliability)、性能(Performance)还有可支持性(Supportability)。使用“FURPS+”模型来实现需求分析可保障用户体验，比较好地能保证系统的运行质量。

以下是对本系统的非功能性需求进行分析：

- a) 易用性：在设计该系统的时候，系统的界面应该整洁美观，按钮名称应该通俗易懂，符合用户的操作习惯，降低用户的学习成本。
- b) 可靠性：系统应该足够鲁棒，能够处理各个模块中可能产生的异常，尽可能避免系统崩溃，如出现不可避免的错误应及时反馈给用户。
- c) 性能：系统应在相关的平台上能流畅运行，系统的时延要能满足一般用户体验，本身应增加与用户交流的对话框，提高用户体验。此外，性能还体现在合成新视图的速度以及质量上，这是本文方法论所主要考量的指标。
- d) 可支持性：系统各功能模块清晰明确，模块之间应该是解耦的，方便日后维护、扩展、更新。接口命名以及教程足够清晰，方便其他开发者使用。

5.1.3 主要用例分析

本小节主要介绍 PC 端的各个应用模块用例。

用户可以进行的操作包括与相机的交互，数据集的处理，以及新视图合成。用例图见图 5-2。

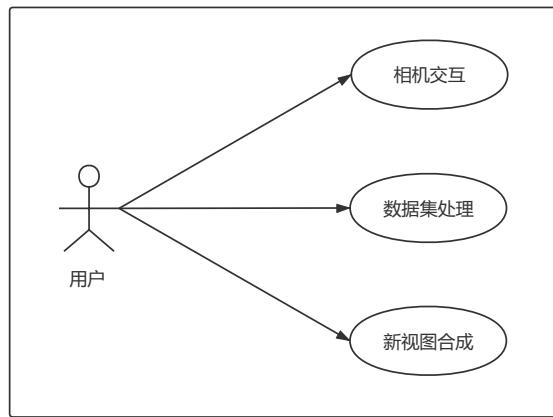


图 5-2 PC 客户端用例图

表 5-1 相机交互用例图 User case#1

用例编号	User case#1
用例名	相机交互
涉众及其关注点	用户：希望进行相机交互功能
前置条件	用户启动应用
主成功场景	<ol style="list-style-type: none"> 1. 用户打开应用，点击“打开相机”按钮 2. 应用跳转至相机界面 3. 用户点击添加按钮 4. 当前相机画面被捕获到数据集栏并显示 5. 用户点击“关闭相机”按钮 <ol style="list-style-type: none"> 5a. 系统提示“您确定要关闭相机吗？” 5b. 系统保存用户处理结果，相机画面终止 6. 用户点击“退出系统”按钮 <ol style="list-style-type: none"> 6a. 系统提示“您确定要退出系统吗？” 6b. 系统保存用户处理结果，整个应用退出
拓展	<ol style="list-style-type: none"> 5a. 相机关闭取消 <ol style="list-style-type: none"> 1. 交互界面保持不变 6a. 系统关闭取消 <ol style="list-style-type: none"> 1. 交互界面保持不变
后置条件	应用显示相应处理结果
发生频率	高

表 5-2 数据集处理用例图 User case#2

用例编号	User case#2
用例名	数据集处理
涉众及其关注点	用户：希望数据集处理功能
前置条件	用户启动应用
主成功场景	<p>1. 用户打开应用，点击“打开相机”按钮</p> <p>2. 应用跳转至相机界面</p> <p>3. 用户点击添加按钮</p> <p>4. 当前相机画面被捕获到数据集栏并显示</p> <p>5. 用户在数据集栏点击“上（下）一张”按钮</p> <p>6. 用户在数据集栏点击“删除”按钮</p> <p>6a. 系统提示“您确定删除吗？”</p> <p>6b. 系统保存用户处理结果，当前图像被删除</p>
拓展	<p>3a. 添加数据集前未打开相机</p> <p>1. 弹出提示框“请打开相机”</p> <p>5a. 数据集为空</p> <p>1. 弹出提示框“请添加图像至数据集”</p> <p>6a. 点击取消删除或提示栏右上角×</p> <p>1. 数据集栏当前画面被保留</p> <p>6b. 删除时，数据集为空×</p> <p>1. 弹出提示框“不可删除空数据”</p>
后置条件	应用显示相应处理结果
发生频率	高

表 5-3 新视图合成用例图 User case#3

用例编号	User case#3
用例名	新视图合成
涉众及其关注点	用户：希望执行新视图合成功能
前置条件	用户启动应用
主成功场景	<ol style="list-style-type: none"> 1. 用户点击“计算位姿”按钮 2. 进度条提示计算位姿的过程 3. 用户点击“训练模型”按钮 4. 进度条提示模型训练的过程 5. 用户点击“合成新视图”按钮 6. 状态提示从“待合成”到“合成中”再到“已合成” 7. 用户在数据集栏点击“上（下）一张”按钮
拓展	<p>1a. 计算位姿时，数据集为空：</p> <ol style="list-style-type: none"> 1. 弹出提示框：“数据集不能为空” <p>3a. 训练模型时，位姿为空：</p> <ol style="list-style-type: none"> 1. 弹出提示框：“请先计算位姿” <p>5a. 训练模型时，未训练模型：</p> <ol style="list-style-type: none"> 1. 弹出提示框：“请先训练模型”
后置条件	应用显示相应处理结果
发生频率	高

5.2 系统设计

本文提出的神经辐射场的新视图快速合成系统主要分为三个部分，包含界面层、业务层、数据层。其中，界面层主要是为用户提供可视化的应用界面以方便用户进行相应的操作。业务层提供了本系统的所有业务基本操作，包括传感器驱动、位姿估计、神经辐射场训练、新视图合成。数据层负责数据库的管理和数据的控制，通过相机获取一系列 RGB 图像，并传给业务层去处理下游任务。本文系统是基于 Windows 平台的应用，对应的 PC 客户端使用 matlab 的 App Design 工具箱实现，具体的系统架构如图 5-3 所示。

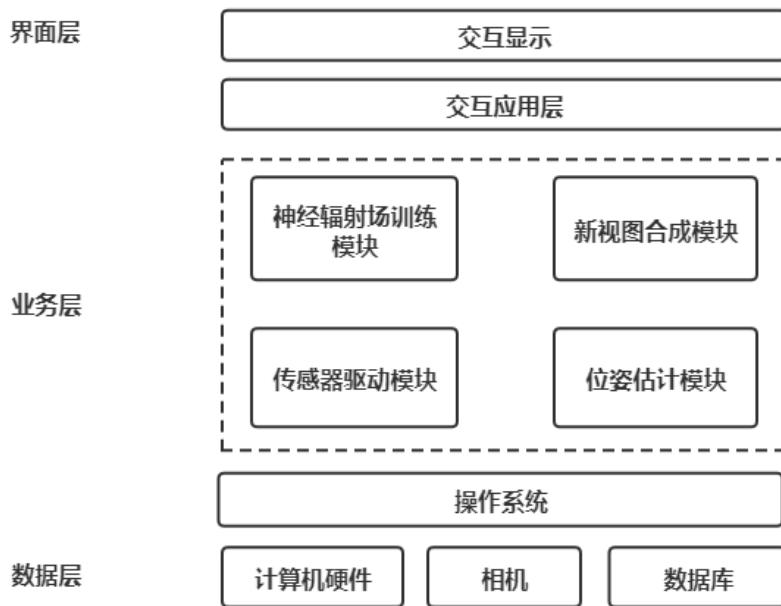


图 5-3 系统功能结构图

5.3 系统实现

本文设计并实现的基于神经辐射场的快速新视图合成系统是基于 Windows 平台开发，算法模块和数据处理模块均是基于 C++ 语言编写，用户交互界面是基于 matlab 语言编写。C++ 和 matlab 语言之间靠动态链接库进行数据通信。由于本文使用的基于神经辐射场的新视图快速合成模型是基于 Pytorch 框架训练的，而移动传感器调用代码使用的则是 C++ 语言，为了联合开发，需要引入 Pytorch 中的 torch.jit API 用来转换成 C++ 语言支持的模型文件上。

数据处理模块的位姿估计模块是基于开源的 COLMAP 框架去估计真实世界

图像的位姿的，当然这不会像合成数据集那样有绝对准确的位姿，质量这一部分势必会下降，但位姿的优化不是本文所研究的问题，在此不作过多阐述。接下来将详细介绍本文快速新视图合成算法模块的 Pytorch 模型是如何移植到 C++ 代码上的，需要使用到 LibTorch 库。

Pytorch 是 Facebook 于 2007 年开发的当今最流行的深度学习框架之一。它有许多优点，比如底层使用 C++ 语言编写，有类似于 numpy 的张量计算，可以用于 GPU 加速，并且支持动态图，编写调试灵活，深受深度学习开发人员的青睐。为了将 pytorch 训练好的模型能够方便地迁移到 C++ 代码中，需要使用 Pytorch 的 C++ API 模型文件 LibTorch。模型转换加载的过程如下：

- 首先是模型转换，创建与模型相同的随机张量，使用 `torch.jit.trace` 模块对运算过程和权值进行跟踪和捕获，进而序列化为 LibTorch 可以加载的模型文件。
- 下载对应平台预编译好的 LibTorch 库，添加 `include` 和 `lib` 目录到编译环境中。
- 在 C++ 代码中加载转换后的 Pytorch 模型即可调用。

```
● ● ●  
1 import torch  
2 import torchvision  
3  
4 # An instance of your model.  
5 model = torchvision.models.resnet18()  
6  
7 """  
8 An example input you would normally provide to your model's  
    forward() method.  
9 """  
10 example = torch.rand(1, 3, 224, 224)  
11  
12 """  
13 Use torch.jit.trace to generate a torch.jit.ScriptModule via  
    tracing.  
14 """  
15 traced_script_module = torch.jit.trace(model, example)
```

图 5-4 Pytorch 模型的转换

```
● ● ●

1 #include <torch/script.h> // One-stop header.
2
3 #include <iostream>
4 #include <memory>
5
6
7 int main(int argc, const char* argv[])
8 {
9     if (argc != 2)
10    {
11        std::cerr << "usage: example-app <path-to-exported-script-
12           module>\n";
13    }
14
15
16 torch::jit::script::Module module;
17
18 try
19 {
20     // Deserialize the ScriptModule from a file using
21     // torch::jit::load().
22     module = torch::jit::load(argv[1]);
23 }
24 catch (const c10::Error& e)
25 {
26     std::cerr << "error loading the model\n";
27     return -1;
28 }
29
30 std::cout << "ok\n";
31 }
```

图 5-5 Pytorch 模型在 C++ 项目中的应用

5.4 本章小结

本章主要介绍了基于神经辐射场的新视图快速合成系统的设计与实现。首先介绍了系统需求分析，包含功能性需求分析和非功能性需求分析，并展示了主要用例分析。最后阐述了此系统的基本设计架构以及将基于神经辐射场的新视图快速合成模型移植到 C++ 代码上的方法。在下一章将会详细介绍本系统的部署与展示。

第六章 系统的部署与展示

本节主要介绍本文系统的部署与测试。本文的系统是运行于 Windows 平台，具体的运行环境见表 6-2。在运行系统之前需要编译和配置好相关运行库，比如 LibTorch、OpenCV、MYNTAI S1040-IR-120/Mono 的 SDK。

6.1 系统运行环境

本系统的开发和运行环境如表 6-1 和表 6-2 所示，包含软件环境和硬件环境。图 6-1 展示的是本系统的真实硬件环境以及相应的系统交互界面。

表 6-1 系统开发环境

软件环境	
操作系统	Ubuntu 18.04
开发语言	python、C++、matlab
运行环境	python 3.6
硬件环境	
处理器	Intel(R) Core(TM) i5-4210U CPU @ 2.50 GHz
内存	8GB
显卡	TITAN X

表 6-2 系统运行环境

软件环境	
操作系统	Windows 10
开发工具	Matlab 2018b
运行库	LibTorch、OpenCV、MYNTAI S1040-IR-120/Mono SDK
硬件环境	
处理器	Intel(R) Core(TM) i5-4210U CPU @ 2.50 GHz
内存	8GB
传感器	MYNTAI S1040-IR-120/Mono

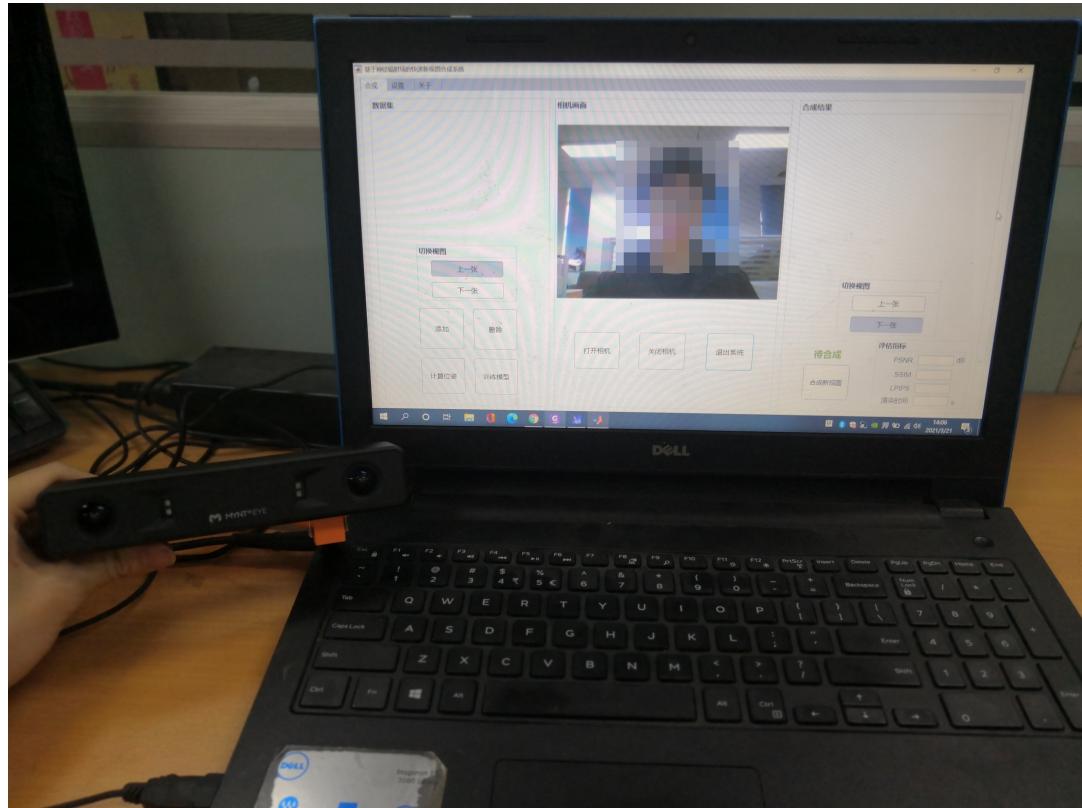


图 6-1 系统的硬件环境

6.2 系统功能测试

本小节针对本系统的各个模块进行测试，下面将详述测试的过程，并展示与用户交互的界面以及最终测试的结果。

6.2.1 相机交互功能

首先我们先测试电脑与相机方面的交互功能。与相机能够友好交互，获得稳定高质量的图像，这对整个训练或者测试都是意义重大地。相机传感器是数据的本源，是一切计算机视觉的基础和灵魂。拥有着一个鲁棒而友好的相机交互环境能为新视图合成任务提供非常好的输入，容易让网络学习到更准确的神经辐射场，预测出更准的颜色和体密度，这些都会在 PSNR、SSIM、LPIPS 等指标上面体现出来。

表 6-3 相机交互功能测试用例

用例编号	001	
测试功能	相机交互	
编号	输入/动作	期望结果
1	打开应用，进入合成界面，在相机画面栏点击打开相机按钮	跳转到传感器 RGBD 相机界面
2	在数据集栏点击添加按钮	当前相机的图像被捕获到数据集栏
3	在相机画面栏点击关闭相机	提示：“您确定要关闭相机吗？”
4	点击确认关闭	相机画面终止
5	点击取消关闭	整个交互界面不变
6	点击退出系统	提示：“您确定退出吗？”
7	点击确认退出	整个交互界面退出
8	点击取消退出	整个交互界面不变

根据表 6-3，整个数据集处理的测试步骤如下：

- 1) 打开应用进入合成菜单栏，点击打开相机按钮，测试结果如图 6-2(a)和图 6-

2(b)。用例 001 编号为 1 的测试与预期结果相符。

- 2) 在数据集面板上点击添加，当前相机画面会被捕获到数据集，如图 6-2(c)所示。用例 001 编号为 2 的测试结果与预期相符。
- 3) 若点击了相机界面的关闭相机按钮，会有相应提示信息，如图 6-2(d)所示。若点击是，则相机画面终止，如图 6-2(e)所示。若点击否，则保持不变，如图 6-2(c)。用例 001 编号为 3、4、5 的测试与预期结果相符。
- 4) 若点击了相机界面的退出系统按钮，会有相应提示信息，如图 6-2(f)所示。若点击是，则程序画面终止。若点击否，则保持不变，如图 6-2(e)。用例 001 编号为 6、7、8 的测试与预期结果相符。



图 6-2 相机交互功能测试

表 6-4 数据集处理功能测试用例

用例编号	002	
测试功能	数据集处理	
编号	输入/动作	期望结果
1	打开应用，进入合成界面，在相机画面栏点击打开相机按钮	跳转到传感器 RGBD 相机界面
2	在数据集栏点击添加按钮	当前相机的 RGB 数据被选作数据集并显示在数据集栏上
3	在数据集栏点击上（下）一张按钮	数据集栏画面切换为上（下）一张图像
4	在数据集栏点击删除按钮	提示：“您确定删除吗？”
5	点击确认删除	数据集栏当前显示的图像被删除
6	点击取消删除或者提示栏右上角 ×	数据集栏当前显示的图像被保留在数据集中

6.2.2 数据集处理功能

数据集处理是整个系统的最先使用的功能，非常至关重要，数据的选取决定了最终测试结果的好坏。数据集处理对应于本系统数据处理模块的功能。首先介绍下数据处理模块对应的典型测试用例，主要包含数据集的获取，删除。

根据表 6-4，整个数据集处理的测试步骤如下：

- 1) 打开应用进入合成菜单栏，点击打开相机按钮，测试结果如图 6-2(b)。用例 002 编号为 1 的测试与预期结果相符。
- 2) 在数据集面板上点击添加，当前相机画面会被捕获到数据集，如图 6-2(c)所示。用例 002 编号为 2 的测试结果与预期相符。
- 3) 若点击数据集栏上（下）一张按钮，视图会切换到上（下）一张图像，如图 6-3(a)和图 6-3(b)所示。用例 002 编号为 3 的测试与预期结果相符。
- 4) 若在数据集栏点击了删除按钮，会有相应提示信息，如图 6-3(c)。若点击是，则数据集当前画面终止，如图 6-3(d)所示。若点击否，则保持不变，如图 6-2(e)。用例 002 编号为 4、5、6 的测试与预期结果相符。

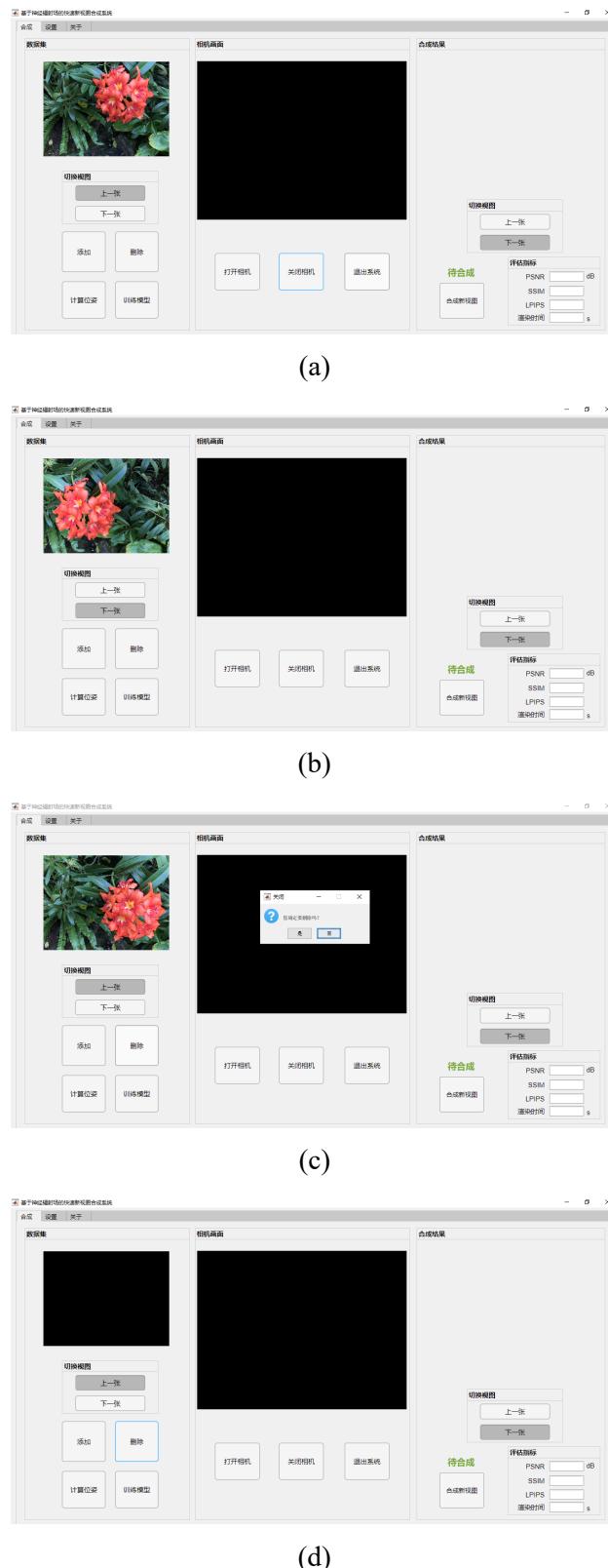


图 6-3 数据集处理功能测试

6.2.3 新视图合成功能测试

基于之前的功能测试，对于本系统的任务来说并不够，因为还没有检验本文方法的可行性与效率。基于神经辐射场的快速新视图合成，最终必须落脚于新视图合成，并对其进行加速。下面是整个系统的最后一部分测试了，是为检测本文整个方法的性能，包含渲染速度以及渲染质量（包含 PSNR、SSIM、LPIPS）等指标。通过多方面去表征本系统能有效地对 NeRF 进行加速。根据表 6-5，整个新视

表 6-5 新视图合成功能测试用例

用例编号	003	
测试功能	新视图合成	
编号	输入/动作	期望结果
1	点击计算位姿按钮	进度条提示计算位姿的过程
2	点击训练模型按钮	进度条提示模型训练的过程
3	点击合成新视图按钮	状态提示从待合成到合成中再到已合成
4	在合成栏点击上（下）一张按钮	数据集栏画面切换为上（下）一张图像，并显示相应的指标信息

图功能测试的详细步骤如下：

- 1) 若点击计算位姿按钮，会有相应提示信息，如图 6-4(a)和图 6-4(b)所示。用例 003 编号为 1 的测试与预期结果相符。
- 2) 若点击训练模型按钮，会有相应提示信息，如图 6-4(c)和图 6-4(d)所示。用例 003 编号为 2 的测试与预期结果相符。
- 3) 若点击合成新视图按钮，会有标签提示进度，如图 6-4(e)和图 6-4(f)以及图6-4(g)。用例 003 编号为 3 的测试与预期结果相符。
- 4) 若点击数据集栏上（下）一张按钮，视图会切换到上（下）一张图像，如图 6-4(h)（上）和图 6-4(g)（下）所示。用例 002 编号为 4 的测试与预期结果相符。



图 6-4 新视图合成功能测试

6.3 本章小结

本章主要介绍了基于神经辐射场的新视图快速合成系统的部署与展示。首先介绍了系统的开发与运行环境。对经典的测试用例进行了详尽的步骤说明，根据用例步骤完成了功能测试。从测试结果看，系统各项功能均能正常运行，合成新视图速度快且质量高。下一章将会对全文进行总结。

第七章 总结与展望

7.1 总结

当前虚拟现实、自由视点视频、街景地图等领域对新视图合成存在大量需求，通过对现有的相关系统进行调研发现存在以下问题：1) 当前市场上的三维渲染的软件难以根据稀疏视图，在任意视角下渲染出高质量的新视图；2) 目前的三维渲染系统大都需要根据深度信息进行三维建模，这需要使用 RGBD 相机，对硬件设备要求较高；3) 目前市场上还没有直接可应用的面向快速新视图合成的交互系统；4) 现有的新视图合成模型需要在空间内进行大量采样，计算时间成本较高，难以完成 PC 端的交互需求。在这样的背景下，本文结合 NeRF 的技术以及查询表缓存加速的相关技术实现了快速合成新视图的神经辐射场渲染具体方法，并设计和实现了一个快速新视图合成的系统，完成了 PC 客户端的相关应用。

本文的主要工作与贡献如下：

- 1) 本文基于 NeRF 预训练的网络预测的体密度信息构建出一个可以表征在物体内外的查询表结构，这一方面直接减少了查询表的尺寸，另一方面该查询表可以直接为空间内任意一点提供 NeRF 网络中间层的特征，这加快了神经网络的正向传播速度。
- 2) 在缓存了上述查询表的基础上，本文改进了 NeRF 的采样过程，通过查询表找到光线上离物体表面最近的内点，使得 NeRF 可以仅在此点附近进行采样，从而不必再使用两个网络，这显著加速了 NeRF。
- 3) 为了补偿查询表分辨率不够带来的精度损失，本文对 F-NeRF 进行再训练，将渲染质量提升到原 NeRF 的水平。
- 4) 本文在公开的数据集上做了大量对比验证实验。在公开合成数据集 Synthetic-NeRF 和真实场景数据集 LLFF-NeRF 上，与 NeRF 相比，我们提出的 F-NeRF 方法分别加速 3.2 倍和 3.4 倍。
- 5) 本文设计并实现了基于 NeRF 的快速新视图合成的系统，完成了相应的 PC 端的应用，并对该系统进行了详细的需求分析以及架构设计，最后将本文的方法实现并部署到了 PC 端，经过详细地测试，该系统满足实际应用需求。

7.2 展望

本文实现的基于神经辐射场的快速新视图合成系统保留了人机交互的自然、简洁等优点，并保证了交互的一致性与实时性，具有一定的应用前景。但系统仍存在一些值得思考与完善的地方：

1. 本文基于 NeRF 改进的快速新视图合成方法本质还是假设输入图像为静态图像，然而在真实场景里，这是很难做到的。因此在未来希望通过改进算法使得系统输入动态图像也可以合成较高质量的视图。
2. 目前本系统的方法使用的查询表提取到的物体几何信息不够准确，这是因为缓存的点过于稀疏，对于空洞较多的场景渲染出的新视图质量不够好。因此在未来可以借助 mesh 缓存场景的几何信息，使用 ray tracing 的方法去找到物体表面，从而可以更好地优化采样点的位置。
3. 由于测试的时候还是使用了神经网络，因此本系统还不足以做到实时的合成新视图。因此未来可以转换思路，从缓存网络参数到直接缓存输出的颜色和体密度，这样可以让测试的过程与神经网络解耦，从而地合成新视图。

参考文献

- [1] Chen S E, Williams L. View interpolation for image synthesis[C]//Whitton M C. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). Anaheim: ACM, 1993:279-288.
- [2] Riegler G, Koltun V. Free view synthesis[C]//Vedaldi A, Bischof H, Brox T, et al. European Conference on Computer Vision (ECCV). Glasgow: Springer, 2020:623-640.
- [3] Kee E, O'Brien J F, Farid H. Exposing photo manipulation with inconsistent shadows[J]. ACM Transactions on Graphics (TOG), 2013, 32:1-12.
- [4] Mildenhall B, Srinivasan P P, Tancik M, et al. Nerf: Representing scenes as neural radiance fields for view synthesis[C]//Vedaldi A, Bischof H, Brox T, et al. European Conference on Computer Vision (ECCV). Glasgow: Springer, 2020:405-421.
- [5] Hedman P, Philip J, Price T, et al. Deep blending for free-viewpoint image-based rendering[J]. ACM Transactions on Graphics (TOG).
- [6] Schonberger J L, Frahm J M. Structure-from-motion revisited[C]//Agapito L, Berg T, Kosecka J, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE Computer Society, 2016:4104-4113.
- [7] Park J J, Florence P, Straub J, et al. Deepsdf: Learning continuous signed distance functions for shape representation[C]//Gupta A, Hoiem D, Hua G, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach: Computer Vision Foundation / IEEE, 2019:165-174.
- [8] Zhang K, Riegler G, Snavely N, et al. Nerf++: Analyzing and improving neural radiance fields[J]. ArXiv, 2020, abs/2010.07492.
- [9] Yen-Chen L, Florence P, Barron J T, et al. inerf: Inverting neural radiance fields for pose estimation[J]. ArXiv, 2020, abs/2012.05877.
- [10] Yu A, Ye V, Tancik M, et al. pixelnerf: Neural radiance fields from one or few images[J]. ArXiv, 2020, abs/2012.02190.
- [11] Trevithick A, Yang B. Grf: Learning a general radiance field for 3d scene representation and rendering[J]. ArXiv, 2020, abs/2010.04595.
- [12] Schwarz K, Liao Y, Niemeyer M, et al. Graf: Generative radiance fields for 3d-aware image synthesis[J]. ArXiv, 2020, abs/2007.02442.
- [13] Liu L, Gu J, Lin K Z, et al. Neural sparse voxel fields[J]. ArXiv, 2020, abs/2007.11571.
- [14] Lindell D B, Martel J N, Wetzstein G. Autoint: Automatic integration for fast neural volume rendering[J]. ArXiv, 2020, abs/2012.01714.
- [15] Qi C R, Su H, Mo K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation[C]//Rehg J, Liu Y, Wu Y, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu: IEEE Computer Society, 2017:77-85.

- [16] Qi C R, Yi L, Su H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space[C]//Guyon I, von Luxburg U, Bengio S, et al. Advances in Neural Information Processing Systems (NeurIPS). Long Beach: Curran Associates, Inc., 2017:5099-5108.
- [17] Yang Y, Feng C, Shen Y, et al. Foldingnet: Interpretable unsupervised learning on 3d point clouds[J]. ArXiv, 2017, abs/1712.07262:5.
- [18] Achlioptas P, Diamanti O, Mitliagkas I, et al. Learning representations and generative models for 3d point clouds[C]//Dy J G, Krause A. International Conference on Machine Learning (ICML). Stockholm: PMLR, 2018:40-49.
- [19] Litany O, Bronstein A, Bronstein M, et al. Deformable shape completion with graph convolutional autoencoders[C]//Forsyth D, Laptev I, Ramanan D, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City: IEEE Computer Society, 2018:1886-1895.
- [20] Baqué P, Remelli E, Fleuret F, et al. Geodesic convolutional shape optimization[C]//Dy J G, Krause A. International Conference on Machine Learning (ICML). Stockholm: PMLR, 2018: 481-490.
- [21] Tarini M, Hormann K, Cignoni P, et al. Polycube-maps[J]. ACM Transactions on Graphics (TOG), 2004, 23:853-860.
- [22] Sinha A, Bai J, Ramani K. Deep learning 3d shape surfaces using geometry images[C]//Leibe B, Matas J, Sebe N, et al. European Conference on Computer Vision (ECCV). Amsterdam: Springer, 2016:223-240.
- [23] Maron H, Galun M, Aigerman N, et al. Convolutional neural networks on surfaces via seamless toric covers.[J]. ACM Transactions on Graphics (TOG), 2017, 36:1-10.
- [24] Groueix T, Fisher M, Kim V G, et al. A papier-mâché approach to learning 3d surface generation[C]//Forsyth D, Laptev I, Ramanan D, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City: IEEE Computer Society, 2018:216-224.
- [25] Ben-Hamu H, Maron H, Kezurer I, et al. Multi-chart generative surface modeling[J]. ACM Transactions on Graphics (TOG), 2018, 37:1-15.
- [26] Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering[C]//Lee D D, Sugiyama M, von Luxburg U, et al. Advances in Neural Information Processing Systems (NeurIPS). Barcelona: Curran Associates, Inc., 2016: 3837-3845.
- [27] Verma N, Boyer E, Verbeek J. Feastnet: Feature-steered graph convolutions for 3d shape analysis[C]//Forsyth D, Laptev I, Ramanan D, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City: IEEE Computer Society, 2018:2598-2606.
- [28] Bruna J, Zaremba W, Szlam A, et al. Spectral networks and locally connected networks on graphs[C]//Bengio Y, LeCun Y. International Conference on Learning Representations (ICLR). Banff: International Conference on Learning Representations (ICLR), 2014.
- [29] Wu Z, Song S, Khosla A, et al. 3d shapenets: A deep representation for volumetric shapes[C]// Grauman K, Learned-Miller E, Torralba A, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston: IEEE Computer Society, 2015:1912-1920.

-
- [30] Choy C B, Xu D, Gwak J, et al. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction[C]//Leibe B, Matas J, Sebe N, et al. European Conference on Computer Vision (ECCV). Amsterdam: Springer, 2016:628-644.
 - [31] Tatarchenko M, Dosovitskiy A, Brox T. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs[C]//Cucchiara R, Matsushita Y, Sebe N, et al. IEEE International Conference on Computer Vision (ICCV). Venice: IEEE Computer Society, 2017: 2107-2115.
 - [32] Riegler G, Osman Ulusoy A, Geiger A. Octnet: Learning deep 3d representations at high resolutions[C]//Rehg J, Liu Y, Wu Y, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu: IEEE Computer Society, 2017:6620-6629.
 - [33] Häne C, Tulsiani S, Malik J. Hierarchical surface prediction for 3d object reconstruction[C]//Avidan S, Yasutaka, Sinha S, et al. International Conference on 3D Vision (3DV). Qingdao: IEEE Computer Society, 2017:412-420.
 - [34] Curless B, Levoy M. A volumetric method for building complex models from range images[C]//Fujii J. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). New Orleans: ACM, 1996:303-312.
 - [35] Jiang C, Sud A, Makadia A, et al. Local implicit grid representations for 3d scenes[C]//Liu C, Mori G, Saenko K, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Seattle: IEEE, 2020:6001-6010.
 - [36] Chabra R, Lenssen J E, Ilg E, et al. Deep local shapes: Learning local SDF priors for detailed 3d reconstruction[C]//Vedaldi A, Bischof H, Brox T, et al. European Conference on Computer Vision (ECCV). Glasgow: Springer, 2020:608-625.
 - [37] Genova K, Cole F, Sud A, et al. Local deep implicit functions for 3d shape[C]//Liu C, Mori G, Saenko K, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Seattle: IEEE, 2020:4857-4866.
 - [38] Mescheder L, Oechsle M, Niemeyer M, et al. Occupancy networks: Learning 3d reconstruction in function space[C]//Gupta A, Hoiem D, Hua G, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach: Computer Vision Foundation / IEEE, 2019: 4460-4470.
 - [39] Chang A X, Funkhouser T, Guibas L, et al. Shapenet: An information-rich 3d model repository[J]. ArXiv, 2015, abs/1512.03012.
 - [40] Niemeyer M, Mescheder L, Oechsle M, et al. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision[C]//Liu C, Mori G, Saenko K, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Seattle: IEEE, 2020:3504-3515.
 - [41] Sitzmann V, Zollhöfer M, Wetzstein G. Scene representation networks: Continuous 3d-structure-aware neural scene representations[C]//Wallach H M, Larochelle H, Beygelzimer A, et al. Advances in Neural Information Processing Systems (NeurIPS). Vancouver: Curran Associates, Inc., 2019:1119-1130.

- [42] Waechter M, Moehrle N, Goesele M. Let there be color! large-scale texturing of 3d reconstructions[C]//Fleet D J, Pajdla T, Schiele B, et al. European Conference on Computer Vision (ECCV). Zurich: Springer, 2014:836-850.
- [43]Debevec P E, Taylor C J, Malik J. Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach[C]//Fujii J. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). New Orleans: ACM, 1996:11-20.
- [44] Wood D N, Azuma D I, Aldinger K, et al. Surface light fields for 3d photography[C]//Brown J R, Akeley K. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). New Orleans: ACM, 2000:287-296.
- [45] Loper M M, Black M J. OpenDr: An approximate differentiable renderer[C]//Fleet D J, Pajdla T, Schiele B, et al. European Conference on Computer Vision (ECCV). Zurich: Springer, 2014: 154-169.
- [46] Chen W, Ling H, Gao J, et al. Learning to predict 3d objects with an interpolation-based differentiable renderer[C]//Wallach H M, Larochelle H, Beygelzimer A, et al. Advances in Neural Information Processing Systems (NeurIPS). Vancouver: Curran Associates, Inc., 2019:9605-9616.
- [47] Genova K, Cole F, Maschinot A, et al. Unsupervised training for 3d morphable model regression[C]//Forsyth D, Laptev I, Ramanan D, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City: IEEE Computer Society, 2018:8377-8386.
- [48] Liu S, Li T, Chen W, et al. Soft rasterizer: A differentiable renderer for image-based 3d reasoning[C]//Kweon I S, Paragios N, Yang M H, et al. IEEE International Conference on Computer Vision (ICCV). Seoul: IEEE, 2019:7708-7717.
- [49] Li T M, Aittala M, Durand F, et al. Differentiable monte carlo ray tracing through edge sampling[J]. ACM Transactions on Graphics (TOG), 2018, 37:1-11.
- [50] Nimier-David M, Vicini D, Zeltner T, et al. Mitsuba 2: A retargetable forward and inverse renderer[J]. ACM Transactions on Graphics (TOG), 2019, 38:1-17.
- [51] Kutulakos K N, Seitz S M. A theory of shape by space carving[J]. International Journal of Computer Vision (IJCV), 2000, 38:199-218.
- [52] Szeliski R, Golland P. Stereo matching with transparency and matting[C]//Davis L, Zisserman A, Yachida M, et al. IEEE International Conference on Computer Vision (ICCV). Bombay: IEEE Computer Society, 1998:517-524.
- [53] Flynn J, Broxton M, Debevec P, et al. Deepview: View synthesis with learned gradient descent[C]//Gupta A, Hoiem D, Hua G, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach: Computer Vision Foundation / IEEE, 2019:2367-2376.
- [54] Mildenhall B, Srinivasan P P, Ortiz-Cayon R, et al. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines[J]. ACM Transactions on Graphics (TOG), 2019, 38:1-14.
- [55] Srinivasan P P, Tucker R, Barron J T, et al. Pushing the boundaries of view extrapolation with multiplane images[C]//Gupta A, Hoiem D, Hua G, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach: Computer Vision Foundation / IEEE, 2019: 175-184.

- [56] Zhou T, Tucker R, Flynn J, et al. Stereo magnification: Learning view synthesis using multi-plane images[J]. ACM Transactions on Graphics (TOG), 2018, 37:65:1-65:12.
- [57] Porter T, Duff T. Compositing digital images[C]//Christiansen H. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). Minneapolis: ACM, 1984: 253-259.
- [58] Sitzmann V, Thies J, Heide F, et al. Deepvoxels: Learning persistent 3d feature embeddings[C]// Gupta A, Hoiem D, Hua G, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach: Computer Vision Foundation / IEEE, 2019:2437-2446.
- [59] Goodfellow I, Bengio Y, Courville A, et al. Deep learning: volume 1[M]. MIT press Cambridge, 2016.
- [60] Hinton G E. Deep belief networks[J]. Scholarpedia, 2009, 4:5947.
- [61] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. Nature, 1986, 323:533-536.
- [62] LeCun Y, Bengio Y, et al. Convolutional networks for images, speech, and time series[J]. The handbook of brain theory and neural networks, 1995, 3361:1995.
- [63] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks[C]//Gordon G J, Dunson D B, Dudík M. International Conference on Artificial Intelligence and Statistics (AISTATS). Fort Lauderdale: JMLR.org, 2011:315-323.
- [64] Krizhevsky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional neural networks[C]//Bartlett P L, Pereira F C N, Burges C J C, et al. Advances in Neural Information Processing Systems (NeurIPS). Lake Tahoe: Curran Associates, Inc., 2012:1106-1114.
- [65] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[C]//Bengio Y, LeCun Y. International Conference on Learning Representations (ICLR). San Diego: International Conference on Learning Representations (ICLR), 2015.
- [66] Srivastava R K, Greff K, Schmidhuber J. Highway networks[J]. ArXiv, 2015, abs/1505.00387.
- [67] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Agapito L, Berg T, Kosecka J, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas: IEEE Computer Society, 2016:770-778.
- [68] Lin H, Xiao Z, Tan Y, et al. Justlookup: One millisecond deep feature extraction for point clouds by lookup tables[C]//Wu J, Mrak M, Li Z, et al. IEEE International Conference on Multimedia and Expo (ICME). Shanghai: IEEE, 2019:326-331.
- [69] Rahaman N, Baratin A, Arpit D, et al. On the spectral bias of neural networks[C]//Chaudhuri K, Salakhutdinov R. International Conference on Machine Learning (ICML). Long Beach: PMLR, 2019:5301-5310.
- [70] Levoy M. Efficient ray tracing of volume data[J]. ACM Transactions on Graphics (TOG), 1990, 9:245-261.
- [71] Lorensen W E, Cline H E. Marching cubes: A high resolution 3d surface construction algorithm[J]. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), 1987, 21:163-169.

- [72] Kingma D P, Ba J. Adam: A method for stochastic optimization[C]//Bengio Y, LeCun Y. International Conference on Learning Representations (ICLR). San Diego: International Conference on Learning Representations (ICLR), 2015.
- [73] Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity[J]. IEEE Transactions on Image Processing (TIP), 2004, 13:600-612.
- [74] Zhang R, Isola P, Efros A A, et al. The unreasonable effectiveness of deep features as a perceptual metric[C]//Forsyth D, Laptev I, Ramantan D, et al. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City: IEEE Computer Society, 2018:586-595.

在学期间完成的相关学术成果

专利:

- [1] 一种基于符号距离函数的三维重建方法: 中国, 2020112279995 (第一发明人)

致 谢

两年的硕士生涯转瞬即逝。回想大四那年，经过慎重考虑，我选择了来外校读研，虽然知道前进道路充满着未知和困难，依旧满怀热情踏上了读研的道路。读研期间有过彷徨，经历过太多迷茫与困难，但是最终也得到了历练。如今毕业在即，回首过去的两年，发现许多人都给予了我很大的帮助。特在此进行由衷的感谢。

首先要感谢我的导师朝红阳教授，本论文的是在她的耐心指导下完成的。朝老师治学严谨，对待科研从来都是一丝不苟。朝老师经常强调做科研要讲究逻辑的严谨性，要把问题定义清楚，认准了方向就要坚持做下去，不要轻言放弃。疫情期间，朝老师坚持线上每周给大家开组会，为大家的科研指明方向。老师始终强调，硕士无论专业硕士还是学术硕士，都要有完整的投稿过程，这极大地促进了实验室同学们科研的积极性。老师的目的是想让他的学生经历科研写作这样一套科学地训练体系，这样以后无论是继续科研还是工作都会有清晰的思路去研究新领域的问题。经过两年的硕士生涯，越发对老师的观点表示强烈认同。

感谢丁圣勇师兄对我的帮助，师兄带我入门 SLAM 领域，通过对 SLAM 的学习，极大地提升了我的工程能力。丁师兄为我的投稿过程操了很多心，每天与我同步工作，指导我修改论文，帮助我完善新视图合成领域的知识体系。同样要感谢图语公司的柯志麟同学和姚王泮同学，在我大四和研一上学期，帮助我入门计算机视觉领域。

感谢实验室的谷溢、仁杰、鸿鑫、泽林同学，大家一起科研的时光很快乐，研究生生涯得到他们的很多帮助，从他们身上学习到很多知识。与谷溢同学合作的 E-NeRF 论文虽然没中，但确实是我们俩以及丁圣勇师兄的心血，很怀念那段时光。谷溢同学对我的科研生涯帮助颇大，耐心帮助我分析实验，调试代码。鸿鑫师兄在我毕业论文写作过程给予了很多帮助，包含如何定义问题，如何写好工程论文，如何保证论文的逻辑严谨性。感谢汇国师兄，牺牲自己的假期时间帮我修改论文的摘要和绪论，让我对文章逻辑的完整性有了深刻的体会。

最后感谢我的女朋友，始终支持我鼓励我，这才让我有了坚持读研的动力。还要感谢我的父母一直以来对我的默默付出。