NeRF 实践

黄宇悦　黄婧媛　吴欣怡　周涛

2024 年 6 月 13 日

摘要

神经辐射场（NeRF）技术是一个利用全连接神经网络（又称多层感知机）的权重表示 3D 场景的框架，可以从部分二维图像中重建复杂的三维场景。我们在基础部分中使用了 NeRF 技术结合 Nerfstudio 中的 Nerfacto 和 Splatfacto 模型，对 Tanks and Temple 数据集中的 M60 坦克视频数据进行了三维重建。DreamGaussian 则是另一种新颖的三维内容生成框架，它是基于一种生成性三维高斯拼接模型，并辅以 UV 空间中的网格提取和纹理细化。在拓展部分中，我们也使用了 DreamGaussian 对一个坦克玩具的二维基础图进行三维重建。

1 技术介绍

1.1 NeRF

神经辐射场（NeRF）[1] 是一个快速发展的研究领域，在计算机视觉、图形、机器人等领域有着广泛的应用。神经辐射场是一个利用全连接神经网络（又称多层感知机）的权重表示 3D 场景的框架，可以从部分二维图像中重建复杂的三维场景。NeRF 已经成为一个受欢迎的研究领域，因为最近的发展扩大了基本框架的性能和能力。最近的发展包括需要更少的图像来训练模型进行视图合成的方法，以及能够从无约束和动态场景表示中生成视图的方法。各种模拟、游戏、媒体应用程序都需要三维图像，此方法用于新视图合成任务，能够从给定的连续视点获得最先进的逼真图像渲染，使数字交互更加逼真和准确。

NeRF 通过构建的一个全连接神经网络，将一个静态场景用一个连续的 5 维函数来表征，这个 5 维的函数在空间中会输出任意位置 (x, y, z) 及任意方向的 (θ, π) 辐照值，同时输出每个点处的一个可微的密度值来表示环境的透光程度。其中全连接神经网络的输入就是一个 5 维坐标 (x, y, z, θ, π)，输出则是一个单一的体密度值以及与视角相关的颜色值。

沿着某个特定的方向进行 NeRF 的渲染的步骤如下：

1. 沿着相机的视角方向生成一组 3D 点。
2. 将这些点和对应的 2D 视角方向输入神经网络，得到一组颜色和密度值。
3. 经典的体渲染技术将这些颜色和密度值渲染得到一张 2D 图像。整个过程都是可微的，所以可以利用梯度下降进行优化，损失函数就是渲染得到的图像与真实图像的差值。

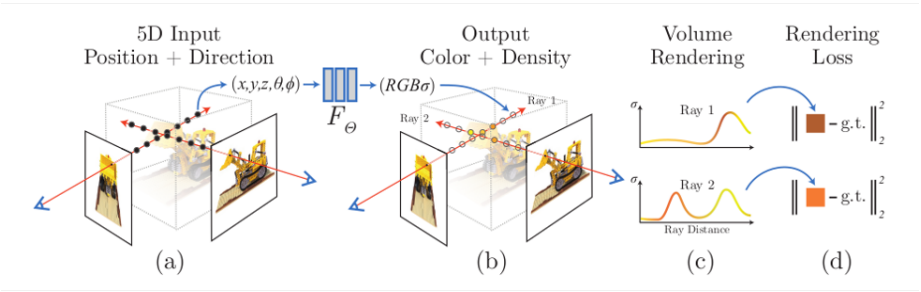

图 1: NeRF 模型

1.2 Nerfstudio

Nerfstudio[2] 是个模块化 PyTorch 框架，用于简化 NeRF 研究的开发和部署。Nerfstudio 中基于 NeRF 的方法的即插即用组件，使得研究人员和从业者可以轻松地将 NeRF 纳入其项目。此外，Nerfstudio 也支持广泛的实时可视化工具、可用于导入野外数据的流线化管道，还包含了用于导出为视频、点云和网格表示的工具。

神经辐射场 (NeRF) 作为一种新兴的三维重建方法，通过训练神经网络从多视角图像中推断出场景的密度和颜色分布，实现高精度的三维重建。然而，传统的 NeRF 在训练效率和渲染速度方面存在瓶颈。为了解决这些问题，我们引入了 Nerfstudio 中的 nerfacto 和 splatfacto 模型。nerfacto 模型在保持高质量重建的同时，通过优化网络结构和训练策略，大幅提高了训练效率。而 splatfacto 模型则利用了点云数据结构，有效提升了渲染速度。在实验中我们通过对 M60 坦克视频数据集的处理和三维重建，我们验证了这些改进模型在实际应用中的有效性和优越性。

1.3 COLMAP

COLMAP[3] 是一个通用的、端到端的基于图像的三维重建管道（即结构-运动（SfM）和多视图立体（MVS）技术），其具有图形和命令行界面。它为有序和无序图像集合的重建提供了广泛的功能，比如图像特征提取、图像特征匹配、场景稀疏重建等。

2 开发环境

我们选择在两种不同环境进行开发。

第一个是在线上使用 Kaggle 以及其提供的 GPU P100 进行开发。

第二个是在本地使用 Windows Subsystem for Linux (WSL) Ubuntu 22.04.3 LTS 进行开发。我们的电脑配置是 AMD Ryzen 4800H CPU 搭配 Nvidia GTX 1650 显卡。

我们跟着 nerfstudio 官网的教程，在这两个环境中安装了 nerfstudio 以及其依赖，比如：PyTorch+CUDA, COLMAP, ffmpeg 等。

```
# 用conda设置名为nerfstudio的新环境
conda create --name nerfstudio -y python=3.10
conda activate nerfstudio
python -m pip install --upgrade pip

# 安装PyTorch和CUDA
pip install torch==2.1.2+cu118 torchvision==0.16.2+cu118
    --extra-index-url https://download.pytorch.org/whl/cu118
conda install -c "nvidia/label/cuda-11.8.0" cuda-toolkit

## 安装Ninja和tiny-cuda-nn
pip install ninja
curl -L
"https://github.com/OutofAi/tiny-cuda-nn-wheels/releases/download/Kaggle-P100/
tinycudann-1.7-cp310-cp310-linux_x86_64.whl" -o
    tinycudann-1.7-cp310-cp310-linux_x86_64.whl
pip install tinycudann-1.7-cp310-cp310-linux_x86_64.whl
    --force-reinstall --no-cache-dir
import tinycudann as tcnn

# 安装nerfstudio
pip install nerfstudio

# 安装COLMAP
conda install -c conda-forge colmap

# 安装ffmpeg
conda install -c conda-forge ffmpeg
```

3 数据介绍

Tanks and Temples 数据集[4] 由英特尔实验室的 Arno Knapitsch 、Jaesik Park 、Qian-Yi Zhou 和 Vladlen Koltun 于 2017 年发布，提供高分辨率的视频。研究人员可以从视频中采集图像，依据图像进行三维重建。此数据集分别提供了训练数据和测试数据，其中测试数据分为中级组和高级组。训练集提供 7 个场景的 7 个高分辨率视频，而测试集共提供 14 个场景的 14 个高分辨率视频。测试集的中级组包含雕塑、大型车辆和具有外观相机轨迹的房屋建筑，而高级组则包含从内部和大型室外场景拍摄的室内场景，其具有复杂的几何布局和相机轨迹。

我们选了 Tanks and Temples 数据集中的 M60 坦克视频作为数据来源。M60 坦克视频数据包含了从不同角度和距离拍摄的图像序列，确保了对坦克的全面覆盖，共计有 5620 帧。同时，M60 坦克视频数据中的图像具有高分辨率，提供了丰富的细节，这对于捕捉坦克的复杂表面和细节结构非常重要，也是我们选取其作为数据的原因之一。

3.1 数据处理

为了使用自定义的数据集进行训练，我们需要将选定的数据集处理成 nerfstudio 能够使用的格式。具体来说，我们需要知道每张图像的摄像机姿势。我们使用了 COLMAP 的图像特征提取、图像特征匹配、场景稀疏重构等功能对其进行处理。整个图像处理过程耗时 22 分钟。

```
# 使用COLMAP对图片进行处理
# 需要提前将原始数据导入WSL root文件夹中
ns-process-data video --data M60.mp4 --output-dir m60/
```

图 2: COLMAP 处理过程和结果

4 模型介绍

4.1 Nerfacto

NeRF Studio 提供了一个调用各种 NeRF 模型的接口，其包含了许多种模型，其中就有 Nerfacto. Nerfacto 由 Nerfstudio 提出，是通过整合多篇论文对 nerf 模型的优化后，所设计出的一个比较全面的神经辐射场模型，也是 Nerfstudio 的默认模型。Nerfacto 模型在保持高质量重建的同时，通过优化网络结构和训练策略，大幅提高了训练效率。

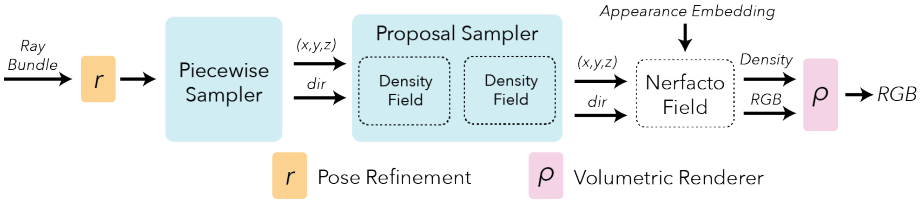

图 3: Nerfacto 模型流水线

上图中的 r 为相机姿态优化（Pose refinement），通过将损失梯度反向传播来修正预测的相机姿态，以减少场景中的模糊和细节损失。而分段采样器（Piecewise Sampler）用于生成初始场景样本。提案采样器（Proposal Sampler）将样本位置限制在对最终渲染贡献最大的区域，从而提高重建质量。最后的密度场 ρ 则是用于引导采样的粗略密度表示。

使用 Kaggle 提供的 P100 GPU 进行模型训练，耗时 43 分钟。

```
# 训练Nerfacto模型
ns-train nerfacto --viewer.websocket-port 7007 --viewer.make-share-url
    True nerfstudio-data --data
    /kaggle/input/tanks-and-temple-m60-colmap-preprocessed/m60
```

4.2 Splatfacto

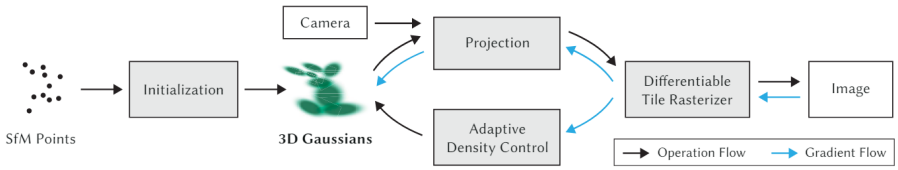

图 4: 3DGS 技术流水线

3DGS（3D Gaussian Splattting)[5] 技术最初由 INRIA 于 SIGGRAPH 2023 提出，具有高质量和实时渲染的能力。由于 NeRF 方法具有需要昂贵的神经网络进行训练和渲染，否则只能牺牲速度以换取质量的缺陷，而 3DGS 则通过了使用 3D 高斯表示场景，优化各向异性协方差以准确表示场景，开发快速的可见性感知渲染算法。因此，3DGS 在 NeRF 的基础上，做到了在保持有竞争力的训练时间的同时实现最先进的视觉质量。

整个过程的起点是一个稀疏的 SfM 点云（SfM Point Cloud), 它可以通过运行结构从运动 (SfM) 算法从多视图输入数据中重建获得。接着，利用 SfM 点云进行初始化 (Initialization), 得到场景的初始 3D 高斯核表示——每个 SfM 点可能会生成一个或多个用于大致描述场景的几何和辐射场特征的高斯核，最终得到一组 3D 各向异性高斯核, 用于编码整个场景的辐射场分布。

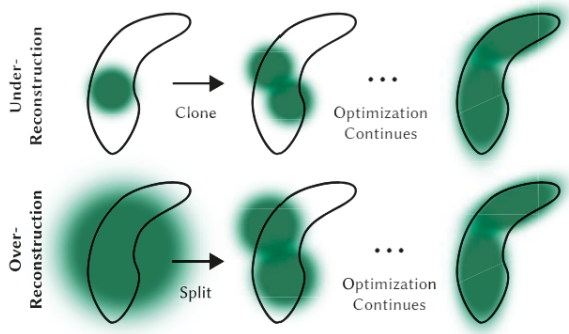

图 5: 自适应密度

接着，使用自适应密度控制策略（Adaptive Density Control）来优化高

斯核的数量和分布。此部分不仅关注几何特征缺失的区域（即重建不足的情况），也关注高斯覆盖场景中大面积的区域（即重建过度的情况），根据重建质量，在某些区域增加或减少高斯核的密度，控制需要填充的空白区域，从而平衡视觉质量和计算效率。将优化后的高斯核集合输入可微分的平铺光栅化器（Differentiable Tile Rasterizer）中，用于生成最终的图像。

为了和 SIGGRAPH 2023 提出的 3DGS 模型作出区分，由 NerfStudio 实现的模型称另成为 SplatFacto，可以理解 Splatfacto 在 3DGS 上做出了更进一步的扩展。如果 Nerfacto 模型是融合多个方法而成的一个模型，则 Splatfacto 模型可以理解为多个高斯溅射法（Gaussian Splatting）融合而成的模型。Splatfacto 通过一系列 3D 体积高斯函数来将给定相机姿态“投影”到 2D 图像上，并进行光栅化以获得每像素颜色。Splatfacto 模型利用了点云数据结构，有效提升了渲染速度。

使用 Kaggle 提供的 P100 GPU 进行模型训练，耗时 27 分钟。

```
# 训练Splatfacto模型
ns-train splatfacto --viewer.websocket-port 7007
    --viewer.make-share-url True nerfstudio-data --data
    /kaggle/input/tanks-and-temple-m60-colmap-preprocessed/m60
```

5 拓展部分

5.1 DreamGaussian 介绍

DreamGaussian[6] 于 ICLR 2024 Oral 提出，是一种新颖的三维内容生成框架，能在短短 2 分钟内完成 Image-to-3D 和 Text-to-3D 任务，即根据文本或 2D 图像重建出高质量的纹理 3D 网格。其主要设计是基于一种生成性三维高斯拼接模型，通过从 3DGS 中提取纹理网格，再辅以 UV 空间中的纹理细化来细致还原物体的几何形状，捕捉其表面的细微纹理和色彩。DreamGaussian 团队证明了三维高斯渐进密集化在三维生成任务中的收敛速度要快得多。

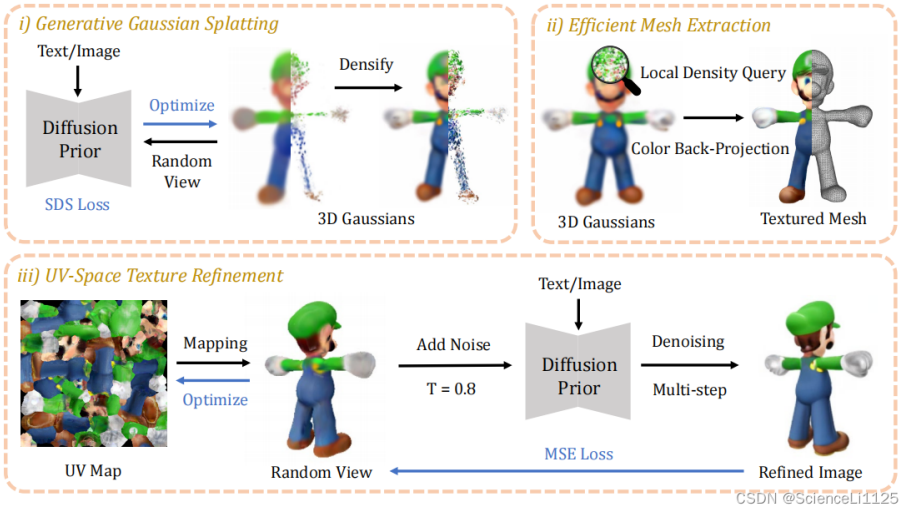

图 6: DreamGaussian 流水线

5.2 数据介绍

我们从互联网上获得了一个坦克玩具的 2D 基础图，并利用 DreamGaussian 对其进行三维重建。

图 7: 坦克玩具 2D 基础图

5.3 代码实现

基于坦克玩具的 2D 基础图，利用 DreamGaussian 进行三维重建。

```
# 安装环境依赖
git clone -b dev https://github.com/camenduru/dreamgaussian

pip install -q torch-ema einops tensorboardX plyfile dearpygui
    huggingface_hub diffusers==0.21.4 accelerate transformers xatlas
pip install -q trimesh PyMCubes pymeshlab rembg[gpu,cli] omegaconf ninja
git clone --recursive
    https://github.com/ashawkey/diff-gaussian-rasterization
pip install -q ./diff-gaussian-rasterization
pip install -q ./simple-knn
pip install -q
    https://github.com/camenduru/wheels/releases/download/colab/nvdiffrast-0.3.1-py3-none-any.whl
pip install -q git+https://github.com/ashawkey/kiuikit

# 导入二维基础图
NAME="tank"
IMAGE="tank.png"
IMAGE_PROCESSED="tank_rgba.png"
run process.py data/{IMAGE}

# 训练DreamGaussian - 第1阶段
run main.py --config configs/image.yaml input=data/{IMAGE_PROCESSED}
    save_path={NAME} elevation=0 force_cuda_rast=True

# 训练DreamGaussian - 第2阶段
run main2.py --config configs/image.yaml input=data/{IMAGE_PROCESSED}
    save_path={NAME} elevation=0 force_cuda_rast=True
```

6 结果展示

基础部分结果

图 8: Nerfacto 模型三维重建和渲染结果

图 9: Splatfacto 模型三维重建和渲染结果

数值评估结果

PSNR（峰值信噪比）是渲染图像与地面实况图像之间峰值误差的测量值，用分贝（dB）表示。PSNR 值越高，表示图像质量越好，与地面实况图像越接近。

数据集	模型	PSNR
M60 坦克数据集	Nerfacto	19.25
M60 坦克数据集	Splatfacto	27.46

拓展部分结果

图 10: DreamGaussian 三维重建和渲染结果

7 项目链接

- GitHub 完整项目仓库（包含模型、代码、动图结果）
- 经过 COLMAP 处理的 M60 坦克数据集
- Nerfacto 模型
- Splatfacto 模型
- Nerfacto 代码
- Splatfacto 代码
- DreamGaussian 代码

8 小组分工

组员	任务
黄宇悦 20300246005	拓展部分 DreamGaussian
黄婧媛 21300246010	基础部分 Nerfacto 和 Splatfacto
吴欣怡 20300246007	撰写实验报告
周涛　21307130239	制作 PPT 和汇报

References

[1] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV].

[2] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. SIGGRAPH ' 23. ACM, July 2023. DOI: 10.1145/3588432.3591516. URL: http://dx.doi.org/10.1145/3588432.3591516.

[3] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[4] Arno Knapitsch et al. "Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction". In: *ACM Transactions on Graphics* 36.4 (2017).

[5] Bernhard Kerbl et al. *3D Gaussian Splatting for Real-Time Radiance Field Rendering*. 2023. arXiv: 2308.04079 [cs.GR].

[6] Jiaxiang Tang et al. *DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation*. 2024. arXiv: 2309.16653 [cs.CV].