基于神经辐射场的纹理合成

摘要

纹理合成是计算机图形学中的一个基本问题，它将有利于各种应用。现有的方法在处理二维图像纹理方面是有效的。相比之下，许多真实世界的纹理在三维几何空间中包含细观结构，如草、树叶和织物，这些纹理不能仅使用二维图像纹理进行有效建模。该论文提出了一种新的纹理合成方法，利用神经辐射场从给定的多视图图像中捕获和合成纹理。在所提出的基于神经辐射场的纹理表示中，具有精细几何细节的场景被分解为细观结构纹理和底层基础形状。这允许将具有细观结构的纹理有效地学习为位于基本形状上的潜在特征，这些特征被馈送到同时训练的神经辐射场解码器中，以表示丰富的视图相关外观。使用这种隐式表示，可以通过潜在特征的补丁匹配来合成基于神经辐射场的纹理。

关键词：神经渲染；纹理合成；微观纹理结构

1 引言

在现实世界中，具有高频几何的纹理无处不在。但用网格、点云或体素来对这种存在着精细几何与复杂拓扑结构的纹理进行建模与合成，虽然直接，但并不是最合适的。而传统的纹理贴图可以表示一系列表面属性，例如颜色、反射、透明度和位移等，但利用传统纹理贴图准确刻画纹理的视角相关外观和精细几何结构仍然不切实际。近些年提出的神经辐射场可以利用拍摄的多视角图像高质量地重建三维场景。但常规的 NeRF 混合了几何与外观的表示，限制了对纹理属性操作的自由度。计算机图形学的应用中，在采集到纹理样本之后，纹理合成是生成相似但不重复的更大纹理以装饰目标表面的一项重要步骤。尽管目前已有大量的工作对二维图像纹理合成进行了研究，但仍缺乏相关工作研究基于神经辐射场的纹理合成。该论文提出的基于神经辐射场的采集、表示、学习与合成具有精细结构和视角相关外观纹理的新颖技术，最终合成得到的纹理还可以应用于任意给定的目标形状，并实现了实时渲染。这也是基于神经辐射场进行纹理合成的首个工作。

2 相关工作

2.1 神经渲染

目前已经提出了各种神经渲染方法来合成具有给定照片集的场景的新颖视图。目前流行的 NeRF 将场景建模为具有发射和阻挡光的粒子的辐射场。受 NeRF 的启发，后续工作对其进行了扩展：其中 Instant-NGP、Plenoxels、ReLU Fields 等实现了比原始 NeRF 更快的重建速度；Nerf++、Mip-NeRF 360、SurfelNeRF、Block-nerf 等实现了大规模场景的重建；DeVRF、NeuPhysics、Nerfplayer 等实现了动态场景的重建；Nerd、NeROIC、NeRV 等实现了光照反射分解的重建；Unified Implicit Neural Stylization、StylizedNeRF、Artistic Radiance Fields 等则实现了风格化效果的重建。目前还提出了一些对细微尺度纹理进行建模的神经表示方法，例如：Kuznetsov 提出的利用神经双向纹理函数对具有细微结构的已知纹理进行建模；NeuTex 通过 UV 参数化在神经表示中明确表示纹理，以支持纹理编辑和映射；NeuMesh 提出了一种基于网格的神经隐式表示方法来解开形状和外观，通过在顶点上定义几何和纹理特征，实现了神经隐式场的几何和纹理编辑。

2.2 纹理合成

早期的纹理合成工作大多是基于传统方法并不断进行改进，1999 年 Efros 和 Leung 等人提出的通过非参数采样进行纹理合成的开创性工作通过逐个分配像素来逐渐扩大合成区域，其中分配是由邻域相似性来决定的。根据这一开创性的想法，在 2000 年 Wei 和 Levoy 等人提出了使用树结构矢量量化的快速纹理合成，使

用了固定邻域以避免不均匀的模式分布。在 2001 年 Lin Liang 等人提出了基于补丁采样的实时纹理合成，提出了在贴片之间混合重叠区域。2001 年 Efros 和 Freeman 等人提出的用于纹理合成和传输的图像填充，以及 2003 年 Kwatra 等人提出的图形剪切纹理:使用图形剪切的图像和视频合成，分别通过动态编程和图形切割来切割纹理重叠区域。2005 年 Kwatra 等人提出了基于实例的合成的纹理优化，这是一种通过纹理优化的替代方法。2013 年 Shi-Min Hu 等人提出的 PatchNet，用于交互式库驱动的图像编辑的基于补丁的图像表示，通过搜索图像库，以定位符合合成约束的理想纹理区域。

除了早期传统的匹配和优化方法以外，神经网络也已经被引入到纹理合成中。2015 年 Gatys 等人提出了使用卷积神经网络的纹理合成，通过优化 VGG 网络提取的潜在的 Gram 矩阵来生成纹理，还有同年 Simonyan 和 Zisserman 等人提出的用于大规模图像识别的甚深卷积网络。在后续的工作中，2016 年 Johnson 等人提出的实时风格转换和超分辨率的感知损失，以及 Ulyanov 等人提出的纹理网络:纹理和样式化图像的前馈合成，通过训练前馈卷积网络以取代耗时的优化过程。生成对抗网络也广泛用于纹理合成中，如 2016 年 Jetchev 等人提出的具有空间生成对抗性网络的纹理合成、2018 年 Yang Zhou 等人提出的对抗性扩展的非平稳纹理合成、2019 年 Hertz 等人提出基于 GAN 架构合成来几何纹理、2020 年 Portenier 等人提出利用 GAN 方法来合成三维纹理、还有 2022 年受 PSGAN 启发的 Haiwei Chen 等人，提出了具有隐式周期场网络的基于示例的模式合成。

3 本文方法

3. 1 方法概述

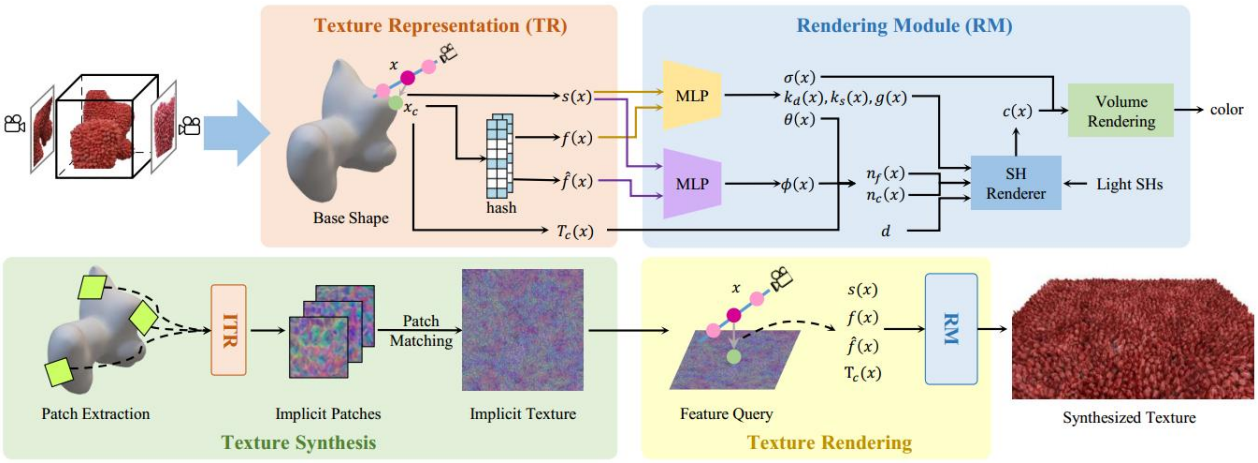

图 1 方法示意图

如图 1 所示，给定一组多视图图像，首先估计其基础形状，对具有细观结构的基础形状和神经辐射场纹理的解纠缠表示进行建模，查询点 x 投影到基础形状上作为基点 x_c 。潜在特征 $f(x)$ 、 $\hat{f}(x)$ 是通过将 x_c 输入进行哈希表中获取的纹理表示。随后将局部切线空间矩阵 $T_c(x)$，潜在特征 $f(x)$ 、 $\hat{f}(x)$，以及 SDF 值 $s(x)$，分别输入到渲染模块的两个多层感知器中。其中密度 $\sigma(x)$，Phong 着色模型系数 $k_d(x)$ 、 $k_s(x)$ 、 $g(x)$，以及精细法线的仰角 $\theta(x)$ 和方位角 $\phi(x)$，基于输入的潜在特征和 SDF 值进行预测。查询点 x 的颜色值 $c(x)$，通过粗法线 $n_c(x)$ 和精细法线 $n_f(x)$，视角方向 d，着色系数 $k_d(x)$ 、 $k_s(x)$ 、 $g(x)$ 以及光照系数 SHs，利用球面谐波(SH)进行计算。在纹理合成阶段，基于隐式纹理表示，从基础形状中提取隐式补丁，并通过隐式补丁匹配算法进行纹理合成。最后通过查询潜在特征，可以从合成的隐式纹理中，重新渲染出合成纹理的外观。

3.2 纹理表示

为了学习微观纹理结构的潜在特征属性，本文将视图中的目标物体分解为微观纹理结构与其所在的粗形状，接着通过在粗形状上定义的潜在特征场将微观纹理结构表示为隐式神经纹理。为实现这一目标，首先使用 Instant-NGP 快速估计场景几何的网格表示，并使用 Co-ACD 进行多凸包分解，过滤高频的微观纹理结构，然后将粗形状重新网格化，使顶点均匀分布在网格表面上，最终得到显式表示粗形状的基础网格。该过程如图 2 所示。

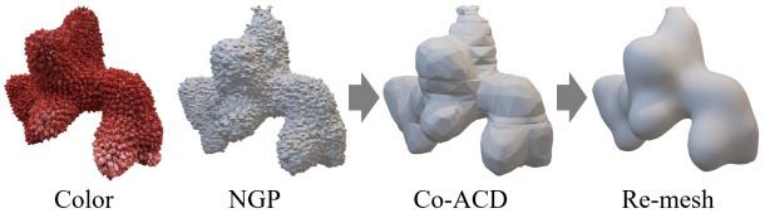

图 2 基础网格提取

随后对表示粗形状的基础网格进投影，获取其表面微观纹理结构的潜在特征。查询点 x 首先被逆着基础网格粗法向 $n_c(x)$ 投影至基础网格表面上，得到投影在基础网格上的基点 x_c 以及带符号距离 $s(x)$，投影过程如图 3 所示。微观纹理结构的潜在征存在基础网格表面定义，并储在哈希表中，通过输入基点 x_c 进行查询与获取。

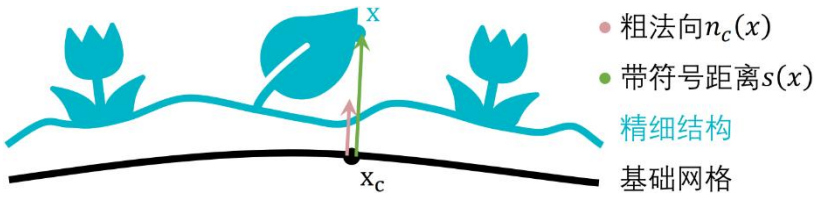

图 3 基础网格投影

3.3 渲染模块

在渲染模块中，局部切线空间矩阵 $T_c(x)$，潜在特征 $f(x)$、$\hat{f}(x)$，以及 SDF 值 s(x)，将分别输入到模块里的两个多层感知器中。其中密度 $\sigma(x)$，Phong 着色模型系数 $k_d(x)$、$k_s(x)$、g(x)，以及精细法线的仰角 $\theta(x)$ 和方位角 $\phi(x)$，基于输入的潜在特征和 SDF 值进行预测。查询点 x 的颜色值 $c(x)$，通过粗法线 $n_c(x)$ 和精细法线 $n_f(x)$，视角方向 d，着色系数 $k_d(x)$、$k_s(x)$、g(x) 以及光照系数 SHs，利用球面谐波(SH)进行计算。

与原始 NeRF 混合了材质和光照的渲染方式不同，本文将着色进行分解，以实现纹理映射到不同模型的实时渲染。同时为了保证实时渲染的速度和稳定收敛，本文使用球面谐波来表示渲染管线中的材质和光照，b 并使用 Phong 着色模型对具有扩散系数 k_d、镜面反射系数 k_s、光泽度 g 这三个参数的材质反射进行建模，最后使用球面谐波来计算纹理颜色 $c(x)$。分解结果如图 4 所示。

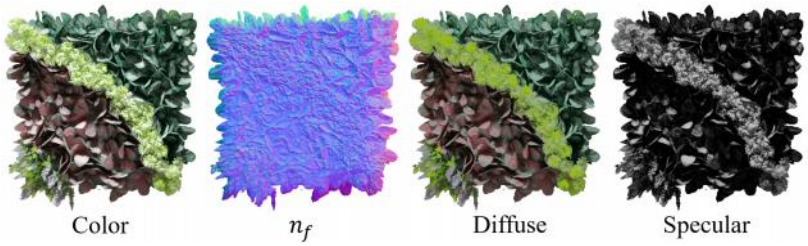

图 4 着色分解结果

3.4 纹理合成

对于采集得到的隐式纹理的贴图应用，将其合成至足够的分辨率是至关重要的步骤。如图 5 所示，通过在定义了潜在特征场的基础网格表面上提取隐式补丁，可以得到隐式补丁的集合。随后通过实施隐式补丁匹配算法，将收集得到的补丁合成为能够实际应用的隐式纹理。在此过程中，算法对潜在特征补丁进行采样、匹配与缝合，合成具有所需分辨率的纹理贴图。此外，本文还引入了一种无监督的度量学习方法来对相似纹理的特征进行聚类，从而提高合成结果的质量。

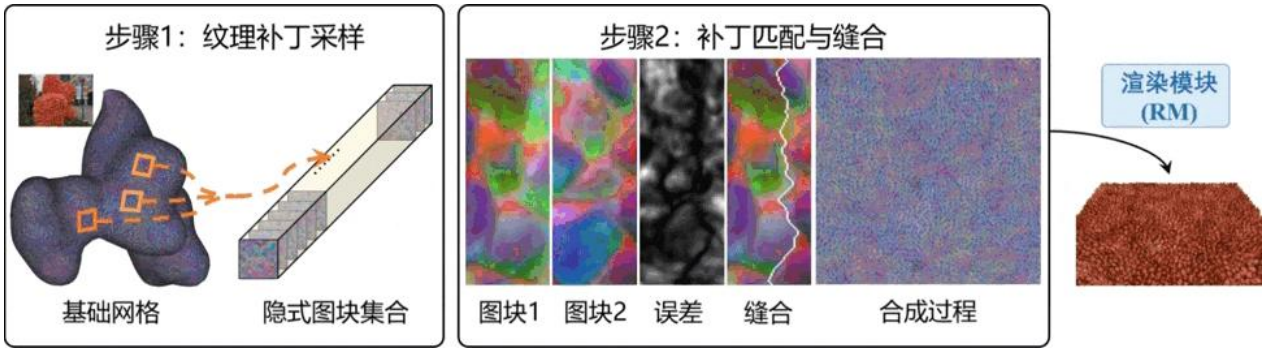

图 5 纹理合成示意图

3.5 模型优化

本文在模型训练中使用 Adam 优化器，损失函数是 L_{rec} 、 $\lambda_1 L_{clu}$ 、 $\lambda_2 L_{dis}$ 、 $\lambda_3 L_{nor}$ 这四项的总和，本文在实际训练时 λ_1 设置为 10^{-5}，λ_2 设置为 10^{-2}，λ_3 设置为 1。其中 L_{rec} 是重建图像与对应视角原始图像的 RGB L1 损失； L_{clu} 是为了避免相似纹理对应的潜在特征，在训练过程中陷入不同最优状态的聚类损失； L_{dis} 是 Mip-NeRF 360 中提出的用于减少重建结果背景塌陷的失真损失；L_{nor} 是使用松弛余弦距离来监督预测法线的法线损失。

4 复现细节

4.1 代码复现情况

论文的源代码是在 torch-npg 开源仓库的基础上进行实现（该仓库是 Instant-NGP 中 SDF 和 NeRF 部分网格编码器、密度网格射线采样器的 Pytorch 实现）。在复现过程中，我主要引用了论文源代码中的基础网格提取相关脚本，如 shape_tools.py、extract_mesh.py、provider.py 等，这些脚本用于从 Instant-NPG 估计的粗形状中利用近似凸包分解等方法提取基础网格；此外还引用了论文源代码中的球谐函数与光照模型相关脚本，如 spherical_harmonics.py、sh_light_model.py、envmap_light_model.py 等，这些脚本是利用球面谐波通过粗法线 $n_c(x)$ 和精细法线 $n_f(x)$，视角方向 d，着色系数 $k_d(x)$ 、 $k_s(x)$ 、 $g(x)$ 以及光照系数 SHs，计算潜在特征的颜色值 $c(x)$。在复现过程中，我的工作主要是根据论文对纹理表示阶段、多层感知机学习阶段、纹理合成阶段的描述，完整地实现了以下功能：查询点在基础网格上的投影、基础网格表面的潜在特征场、多层感知机网络及其学习过程、以及隐式纹理块的采样、隐式纹理的缝合过程；并按照作者所使用的训练过程和训练参数，对模型进行训练，尽可能地接近作者的训练结果和合成效果。

4. 2 实验环境搭建

实验运行环境所需的第三库(可以自行搜索安装)：

Python 3.7+、PyTorch(1.11.0+cu113)、PyTorch3D、Tiny-CUDA-NN、PyMesh、FRNN、cuBVH

实验运行环境所需的内置库(根据项目源码安装)：

Raymarching、Raytracing、GridEncoder、requirements.txt

编译并安装实验所需的第三方工具(根据各自的项目仓库安装)：

CoACD: 是具有碰撞感知凹度和树搜索的 3D 网格的近似凸分解论文的开源实现，该工具用于实验的基础网格提取阶段，对 Instant-NPG 估计的粗形状进行凸包分解，过滤高频的微观纹理结构。

Manifold: 是 ShapeNet 模型的鲁棒水密流形表面生成方法论文的开源实现，该工具同样用于实验的基础网格提取阶段，对凸包分解后的粗形状，重新进行网格化，以获得更加平滑的基础网格。

MiVOS: 是模块化交互式视频对象分割论文的开源实现，该工具用于实验的数据准备阶段，通过交互式分割可以从拍摄的场景视频中获取所需物体的多视图数据，该数据用于后续的粗形状估计。

以上所有第三方工具编译安装完成后，将指定的程序文件放入指定的文件夹路径中。

4. 3 界面使用说明

在数据准备阶段，运行 prepare_your_data.py 脚本后，将出现如图 6 所示的交互界面。第一步，在主视图画面中左键点击需要分割的区域；第二步，在主视图画面中右键点击不需要保留的区域；第三步，点击界面右下角的“Propagate”按钮，将分割结果传播至视频的每一帧；第四步，点击左下角的“Play”按钮，播放视频检查分割结果，第五步，检查完毕再次点击按钮停止播放；第六步，点击右下角的“Save”按钮，保存最终的分割结果；完成上述步骤后即可关闭交互界面。

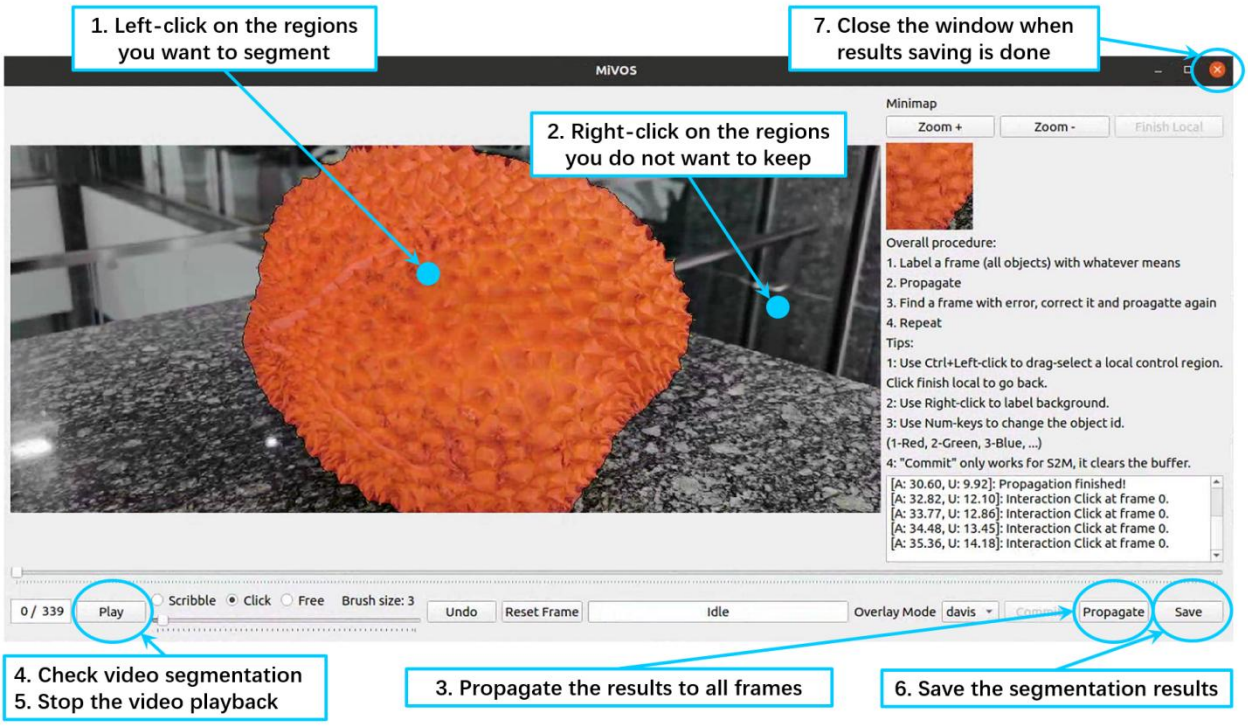

图 6 数据准备界面使用

在模型训练阶段，运行 python main.py 脚本后，将出现如图 7 所示的交互界面。第一步，点击“Start”按钮，开始训练学习基础网格表面的潜在特征，大约在 10-20 分钟左右后，模型将得到较好的收敛效果，如果是一般的显卡则需要更长训练的时间；第二步，当模型收敛时，点击“Stop”按钮，并点击“Save checkpoint”按钮以保存训练好的模型；第三步，若需要可对基础网格选取指定的采样区域；第四步，点击“Sample patches”按钮对基础网格进行隐式纹理采样；第五步，运行 patch_matching_and_quilting.py 脚本，利用采样的隐式纹理块进行纹理合成；第六步，点击“load synthesis”按钮，加载并显示所合成的纹理；第七步，点击“load_shape”按钮，将合成的纹理应用到指定路径的模型中。

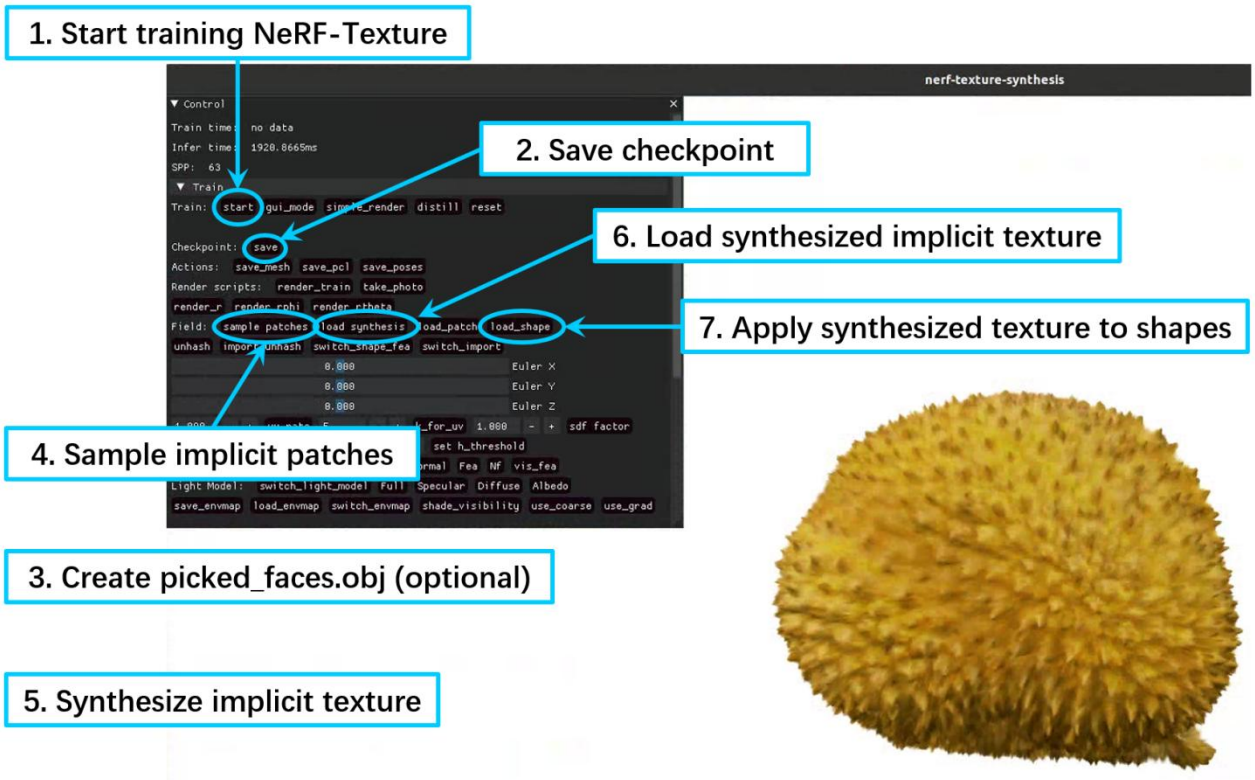

图 7 模型训练界面使用

5 实验结果分析

以下是本次复现实验的结果，并将自己的训练结果，与作者的训练结果进行对比。如图 8-10 所示，本次复现实验的结果，在整体上较为接近作者的训练结果，纹理结构也基本符合输入视图的纹理样式。但是在纹理细节上，还是不如作者的训练结果，自己训练的结果存在割裂、模糊的情况，且边缘存在一些漂浮物。可能是复现代码存在一些缺陷，也可能是训练条件存在差距。

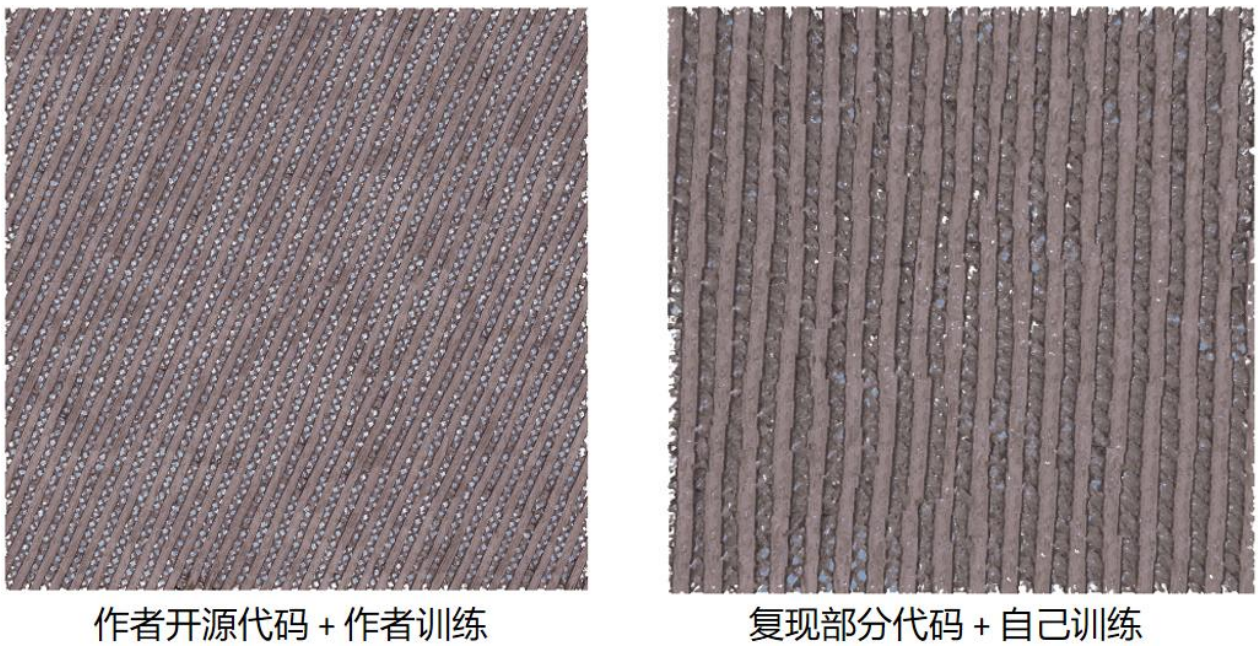

图 8 训练结果对比图 1

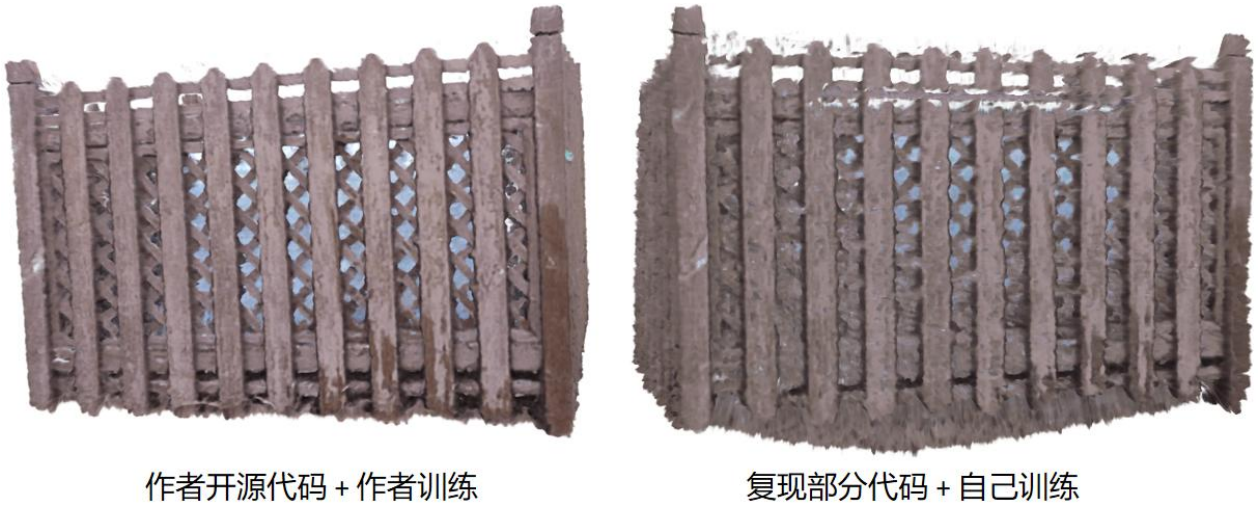

图 9 训练结果对比图 2

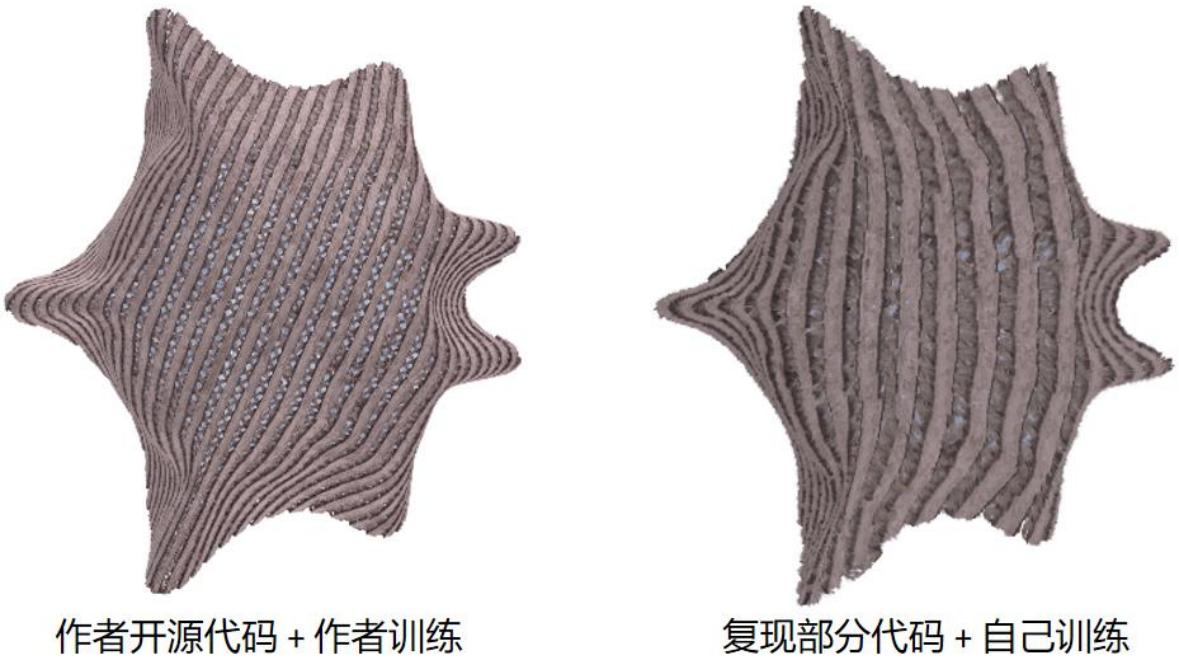

图 10 训练结果对比图 3

6 总结与展望

本报告的内容是关于基于神经辐射场的纹理合成，首先在相关工作部分描述了神经渲染以及纹理合成的经典工作以及近几年较出色的成果；接着对论文所使用的方法以及各个模块进行了描述，该论文的主要核心模块是表示隐式纹理的潜在特征、学习隐式纹理并分解着色的渲染模块、采样和缝合隐式纹理块的纹理合成模块；随后讲述了本次的复现过程，包括代码复现、环境搭建、以及界面使用；最后展示了本次复现的效果。本次复现效果已经尽可能地接近作者的训练效果，但在细节上还是不如作者的训练效果，可能是复现代码存在一些缺陷，也可能是训练条件存在差距。以后可以尝试改进代码并用更好的资源进行复现。后续还可以使用最新的新视图合成技术，如 3D Gaussian Spaltting 等，来进一步研究提升微观纹理结构的重建与合成效果。

参考文献

[1] Y.-H. Huang, Y.-P. Cao, Y.-K. Lai, Y. Shan, and L. Gao,“NeRF-Texture : Texture synthesis with neural radiance fields”in Proc. SIGGRAPH Conf. Papers, 2023, Art. no. 43.