高保真度的神经表面重建

摘要

神经表面重建已被证明可用于通过基于图像的神经渲染恢复密集的 3D 表面。然而，目前的方法很难恢复真实场景的详细结构。为了解决这个问题，我们提出了 Neuralangelo [3]，它将多分辨率 3D 哈希网格的表示能力与神经表面渲染相结合。两个关键技术使我们的方法成为可能：(1) 用于计算高阶导数的数值梯度作为平滑操作，(2) 对控制不同级别的细节的哈希网格进行从粗到细的优化。即使没有深度等辅助输入，Neuralangelo 也可以有效地从多视图图像中恢复密集的 3D 表面结构，保真度显著超过了以前的方法，从而能够从 RGB 视频捕获中进行细粒度的大规模场景重建。

关键词： 3D 哈希网格；高阶导数；3D 表面重建

1 引言

3D 表面重建旨在从在不同视点观察到的多幅图像中恢复密集的几何场景结构。恢复的表面为许多下游应用程序提供了有用的信息，例如用于增强/虚拟/混合现实的 3D 资产生成或用于机器人导航的环境映射。传统的多视角三维重建方法面临着许多缺陷，其难以处理模糊的预测，比如说当碰到重复的纹理或者剧烈的颜色变化，传统方法会显著的降低重建的质量。最近神经辐射场 NeRF [5] 的提出，展现出了神经表面重建方法在解决传统方法上面的优势。通过多层 MLP 来展现该 3D 场景为一个隐式函数，如 SDF 函数。且利用了 MLP 内在的连续性和神经体渲染公式，这些技术允许优化的表面在空间位置之间有意义插值，从而产生平滑和完整的表面表示。此方法更是结合了新的具有可拓展性的 Instant-NGP [6] 的表示，以及多尺度哈希编码方式的应用，使得模型中的 MLP 网络更具表现能力，重建效果更好。

2 相关工作

三维重建最开始起源于基于传统的 MVS 方法，从多视角图像中获得深度图再进行融合得到对应的 3D 点云。在这之后传统方法迎来改进，逐步以增量式重建的 Colmap [7] 为主。随着神经辐射场 NeRF 的爆火，现在的主流方法基本都是基于 NeRF, 结合相应的 3D 表面表示来对多视角图像进行重建。

2.1 多视角表面重建

早期的基于图像的摄影测量技术使用体积占用网格（既体素）来表示场景。如果满足相应投影图像像素之间的严格颜色规定，则访问每个体素并标记占用。该方法的缺陷在于体素所

受限的分辨率，故难以去恢复细粒度的几何细节。后续方法通常从多视图立体几何的 3D 点云开始，然后进行密集表面重建。但是依赖生成的点云的质量通常会导致缺失或嘈杂的表面。比如有一些常见的通过多视角关系进行重建的方法 [8] [9]

2.2 神经辐射场 (NeRF)

该论文中提出了一种方法，通过使用稀疏的输入视图集，优化底层的连续体积场景函数，来将复杂场景的不同的视图进行合成。论文中提出的算法使用全连接神经网络（非卷积神经网络）表示场景，其输入是单个连续的 5D 坐标（空间位置 (x，y，z) 和观察方向 (θ，Φ)，其输出是该空间位置处的体积密度和与视角相关的光线信息。我们通过查找某一束相机光线（实际上是物体表面反射进入镜头的光线）上若干采样点的 5D 坐标来合成视图，并使用经典的立体渲染技术将输出的颜色和密度投影到图像中。通过对所有视角下采样的信息的融合，就可以很好的生成该物体在任意视角下的图片信息。本论文的工作能很好解决将对象和场景表示为连续函数的工作，并且提出体素采样的方法对于三维重建有着巨大的作用。但对于三维物体的重建工作，论文中提出的模型并没有对图片中的表面信息进行细节化的处理，所以重建出来的模型外观信息将会有一定的丢失。此外，NeRF 及其变体的一个问题是如何定义体积密度等值面来表示底层 3D 几何的问题。目前的实践通常依赖于对密度值的启发式阈值; 然而，由于不同层级下的约束不足，这样的表面通常是有噪声的，可能无法准确地对场景结构进行建模。因此，NeRF 相比于重建来说更适用于新视角生成。具体流程如下图 1所示

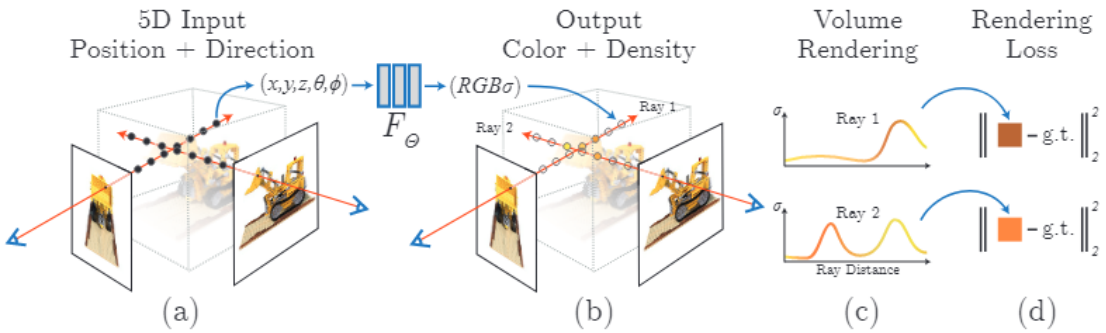

图 1. Nerf 的具体流程

2.3 神经表面重建 (NeuS [10])

对于具有更好定义的 3D 表面的场景表示，像占用网格或 SDF 这样的隐式函数优于简单的体积密度场。为了与神经体渲染结合，人们提出了不同的技术，将底层表示重新参数化回体积密度。这些神经隐函数的设计使表面预测更加准确。该论文主要目标就是重建得到相应物体三维网格。输入是多视角 RGB 待重建物体图片，而输出就是经过网络得到待重建场景关于待重建物体的 SDF 函数以及该物体的高精度三维网格；同样也可以得到新视角渲染图片结果。这可以通过 Marching Cube [4] 算法作用在 SDF [2] 场上得到，因此如何得到 SDF 则成了关键，NeuS 很关键的一点就是将渲染重建优化与 SDF 网络训练关联在一起，联合优化出视感优秀的 SDF 场。传统 NeRF 只关注了新视角合成 (Novel View Synthesis)，涉及到物体三维网格提取则效果不理想，会有很多游离的噪声点存在；而 IDR [13] 等方法虽然是用于重建表面的，但其所用渲染方式是传统的表面渲染，对于表面深度会发生突变的物体无法重

建得很好，很容易就会优化到一个局部最优解，与真实 GT 结果大相径庭。NeuS 最大的贡献就是参考了传统体渲染进行三维物体表面 SDF 重建，并在一些策略上更改了传统的体渲染公式，以此解决了传统体渲染公式中权重函数 W(t) 与体密度 (t) 达到最值的时刻不一致导致在物体重建中出现残影的问题。这种改变提高了表面重建网格的准确性，同时也反馈于新视角合成使得重建结果更加鲁棒，效果更好。

3 本文方法

3.1 本文方法概述

Neuralangelo 从多视图图像重建场景的密集结构。Neuralangelo 沿相机视图方向采样 3D 位置，并使用多分辨率哈希编码来编码位置。编码的特征被输入到 SDF MLP 和颜色 MLP，使用基于 SDF 的体渲染方程来生成图像。并且通过计算高阶导数的数值梯度和引入曲率损失作为平滑操作。

3.2 数值梯度的计算

哈希编码的位置分析梯度计算存在局部性。因此，优化更新只传播到局部哈希网格，缺乏全局平滑性。如图 2所示，为了确保表面表示的一致性，需要对这些网格单元进行联合优化。然而，分析梯度仅限于局部网格单元，除非所有相应的网格单元碰巧同时采样和优化。这种抽样并不总是得到保证。为了克服哈希编码分析梯度的局部性，我们建议使用数值梯度计算表面法线。如果数值梯度的步长小于哈希编码的网格大小，则数值梯度等价于分析梯度; 否则，多个网格单元的哈希编码将参与表面法线计算。因此，通过表面法线反向传播允许多个网格的散列条目同时接收优化更新。直观地说，具有精心选择的步长的数值梯度可以解释为对分析梯度表达式的平滑操作。

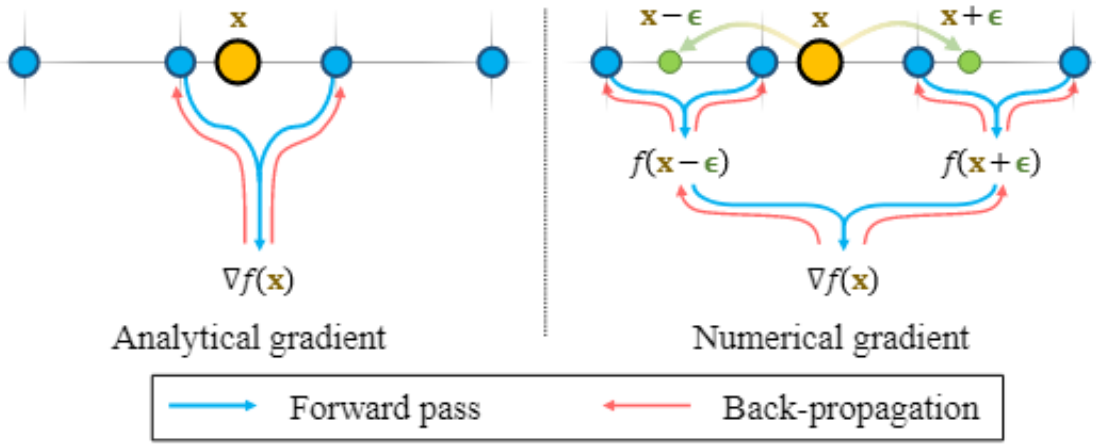

图 2. 使用数值梯度进行高阶导数将反向传播更新分布在局部哈希网格单元之外，从而成为分析梯度的平滑版本。

3.3 多分辨率哈希编码

其渐进式逐层恢复细节如图 3所示

步长 ε

步长 ε 控制分辨率和恢复细节的数量。在实践中，我们将步长 初始化为最粗的哈希网格大小，并在整个优化过程中匹配不同的哈希网格大小以指数方式减小它。在数值表面法线计算中施加较大的 ε 来确保表面法线在更大范围内保持一致，从而产生一致且连续的表面。另一方面，施加较小 ε 的 L_{eik} 会影响较小的区域并避免平滑细节。

哈希网格分辨率 V

我们仅启用一组初始的粗散列网格，并在整个优化过程中当 ε 减小到其空间大小时逐步激活更精细的散列网格。在实践中，我们还在所有参数上应用权重衰减，以避免单分辨率特征主导最终结果。

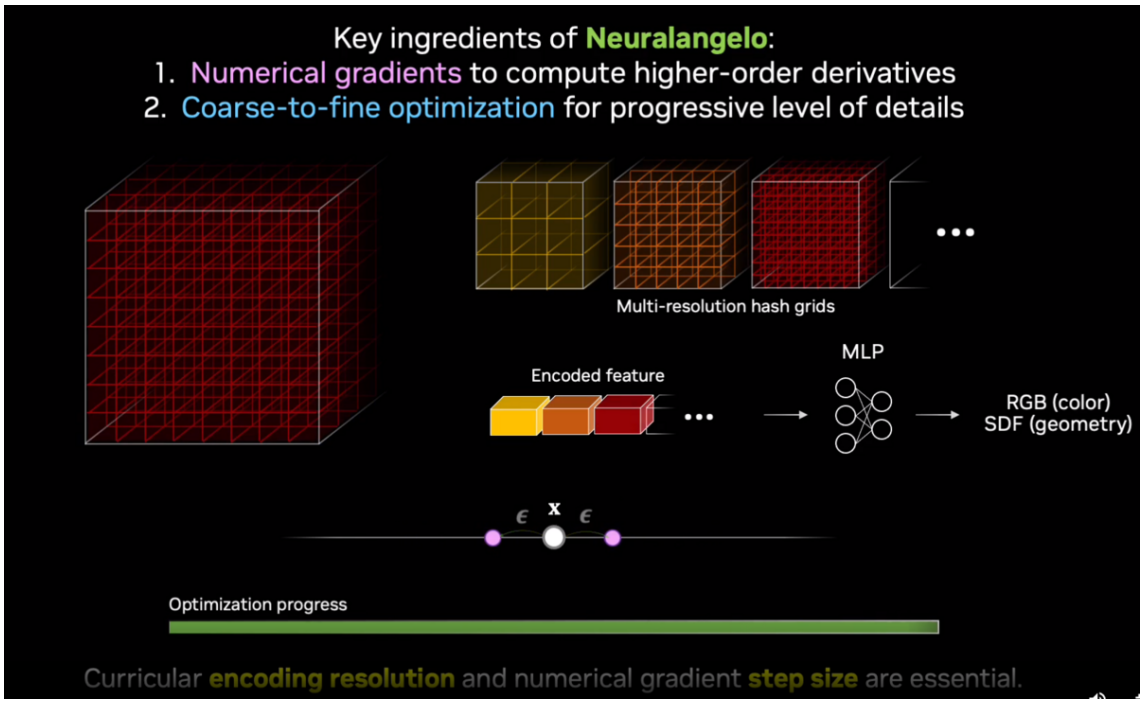

图 3. 渐进式哈希编码

3.4 损失函数定义

为了进一步鼓励重构曲面的光滑性，我们通过正则化 SDF 的平均曲率来施加先验。曲率损失 L_{curv} 如下所示：

$$L_{curv} = \frac{1}{N}\sum_{i=1}^{N}|\nabla^2 f(x_i)|$$

而模型的总损失 Loss 定义如下所示:

$$L = L_{RGB} + w_{eik}L_{eik} + w_{curv}L_{curv}$$

4 复现细节

4.1 与已有开源代码对比

复现过程中引用参考了本文作者在 git-hub 上开源的代码，采用了其定义的神经隐式表达方式以及其模型的定义。在它原本采用的数据集上进行更新，不再采用所占区域更大的 DTU 标准数据集，而是采用了在 23 年发表的一篇论文中的 Omint3D 数据集，其数据集相比于 DTU 中的物体所占图像的区域更小，拥有更丰富的背景，而且类别更加不同。还在它代码的基础上进行了创新，在其代码中应用了基于 mask 的损失函数来利用数据集中的 mask 进行监督，以及根据 colmap 所得到稀疏的点云对其 SDF 函数进行一个显式的监督来得到更加拟合的 SDF 函数。改进的代码在结果上相比于原始代码去除了多余的背景，重建效果更好。

4.2 实验环境搭建

本实验环境在服务器上进行搭建，主要是先下载该源码文件，通过该链接Neuralangelo下载，下载后解压至你所需要运行的文件目录下，然后直接在 anaconda 中运行如下两条命令即可搭建好该环境。

```
conda env create --file neuralangelo.yaml
conda activate neuralangelo
```

4.3 界面分析与使用说明

4.3.1 模型训练

```
EXPERIMENT=toy_example
GROUP=example_group
NAME=example_name
CONFIG=projects/neuralangelo/configs/custom/${EXPERIMENT}.yaml
GPUS=1  # use >1 for multi-GPU training!
torchrun --nproc_per_node=${GPUS} train.py \
    --logdir=logs/${GROUP}/${NAME} \
    --config=${CONFIG} \
    --show_pbar
```

4.3.2 抽取表面

```
CHECKPOINT=logs/${GROUP}/${NAME}/xxx.pt
OUTPUT_MESH=xxx.ply
CONFIG=logs/${GROUP}/${NAME}/config.yaml
RESOLUTION=2048
BLOCK_RES=128
GPUS=1  # use >1 for multi-GPU mesh extraction
```

```
torchrun --nproc_per_node=${GPUS} \
    projects/neuralangelo/scripts/extract_mesh.py \
    --config=${CONFIG} \
    --checkpoint=${CHECKPOINT} \
    --output_file=${OUTPUT_MESH} \
    --resolution=${RESOLUTION} \
    --block_res=${BLOCK_RES}
```

4.4 创新点

主要是新数据集和采用的 Mask-loss 与 SDF-loss 两个损失函数用于监督网络的训练，使得重建效果更好。

4.4.1 新数据集

比较基于两者数据集的不同，原本 Neuralangelo 用的物体数据集为 DTU [1] 数据集，而我们现在用的 OmniObject3D [11] 数据集相比于 DTU 具有更丰富的背景信息，况且所占图片的区域更小，对模型针对于小物体重建的能力提出了挑战。

图 4. OmniObject3D 数据集

4.4.2 Mask 损失

首先先采用图 5代码对 mask 进行预处理，读入 mask 并与代码中的颜色贡献概率度 (opacity) 类比于 Neus 代码中做交叉熵损失进行计算。

```python
    # 获得每张图片的mask及其尺寸大小
    2 usages
    def get_mask(self, idx):
        # 获得每张图片的路径
        fpath = self.list[idx]["file_path"]
        # 转化为mask的路径
        fpath = fpath.replace("jpg", "png")
        fpath = fpath.replace("images", "segs")
        mask_fname = f"{self.root}/{fpath}"
        mask = Image.open(mask_fname)
        mask.load()
        # 得到尺寸大小
        mask_size_raw = mask.size
        return mask, mask_size_raw

    # 预处理mask，进行裁剪并得到裁剪后的图片的mask值
    1 usage
    def preprocess_mask(self, mask):
        # Resize the image.
        mask = mask.resize((self.W, self.H))
        mask = torchvision_F.to_tensor(mask)
        mask = mask[:1]
        return mask
```

图 5. 对 mask 进行预处理

如图 6所示，将 mask 的损失加入训练阶段的损失中

```python
# 计算损失
def _compute_loss(self, data, mode=None):
    # 训练数据的损失
    if mode == "train":
        # Compute loss only on randomly sampled rays.
        self.losses["render"] = self.criteria["render"](data["rgb"], data["image_sampled"]) * 3  # FIXME:sumRGB?!
        self.metrics["psnr"] = -10 * torch_F.mse_loss(data["rgb"], data["image_sampled"]).log10()
        if "eikonal" in self.weights.keys():
            self.losses["eikonal"] = eikonal_loss(data["gradients"], outside=data["outside"])
        if "curvature" in self.weights:
            self.losses["curvature"] = curvature_loss(data["hessians"], outside=data["outside"])
        # 添加mask的loss用于监督
        if "mask" in self.weights:
            self.losses["mask"] = mask_loss(data["opacity"], data["mask_sampled"], loss_type: "mse")
```

图 6. 添加 mask 的损失

4.4.3 SDF 损失

根据 Geo-NeuS [12] 的模型架构来对我们的模型进行优化。首先得通过我们传统的 Colmap 软件来得到对应物体的稀疏点云，输入多视角图片，并且在进行完特征提取，特征匹配，图像校准进行增量式重建后得到的稀疏点云，其中可以导出 sparse 文件，并且使用 sparse 文件内的 images.bin, points3D.bin。images.bin 文件中保存的是对应视角图像下可以看见的点的索引和其图像坐标，points3D.bin 文件中保存的是整个场景中所有的稀疏点的索引以及其世界坐标系下的坐标。首先我们需要对其进行处理从而得到我们在每个视角图像下可以看见的稀疏点云, 比如得到对应的 points.npy 和对应的 view-id.npy 文件，如图 7, 8所示

```
# get the sfm points
pts_dir = f"{cfg.data.root}/sfm_pts/points.npy"
view_id_dir = f"{cfg.data.root}/sfm_pts/view_id.npy"
self.pts = torch.from_numpy(np.load(pts_dir)).cuda()
self.pts_view_id = np.load(view_id_dir, allow_pickle=True)
if cfg.model.object.sdf.encoding.type == "hashgrid" and cfg.model.object.sdf.encoding.coarse2fine.enabled:
    self.c2f_step = cfg.model.object.sdf.encoding.coarse2fine.step
    self.model.module.neural_sdf.warm_up_end = self.warm_up_end
```

图 7. 得到稀疏点云

```
# add mask loss
if "mask" in self.weights:
    self.losses["mask"] = mask_loss(data["opacity"], data["mask_sampled"], loss_type: "mse")
    self.writer.add_scalar( tag: "mask_losss", self.losses["mask"].detach(), self.current_epoch)
# add sdf loss
if "sdf" in self.weights:
   batch_idx = data["idx"]
   idx_0 = batch_idx[0]
   idx_1= batch_idx[1]
   pts_view_0 = gen_pts_view(self, idx_0)
   pts_view_1 = gen_pts_view(self, idx_1)
   pts_view = torch.cat( tensors: (pts_view_0, pts_view_1), dim=0)
   pts2sdf = self.model.module.neural_sdf.sdf(pts_view)
   self.losses["sdf"] = torch_F.l1_loss(pts2sdf, torch.zeros_like(pts2sdf), reduction='sum') / pts2sdf.shape[0]
   self.writer.add_scalar( tag: "sdf_losss", self.losses["sdf"].detach(), self.current_epoch)
```

图 8. 添加 SDF 的损失

5 实验结果分析

本部分对实验所得结果进行分析，与基准方法对比，如图 9所示, 左边的为基准既源代码所跑出来的结果，而右边的是我们用我们修改后的代码所跑出的结果。从对比实验结果发现我们的方法生成的结果更加细致，且我们不再生成背景那些复杂的不感兴趣的区域，反而集中于我们的物体的重建。此外我们的重建效果也更加平滑，整体质量更高。

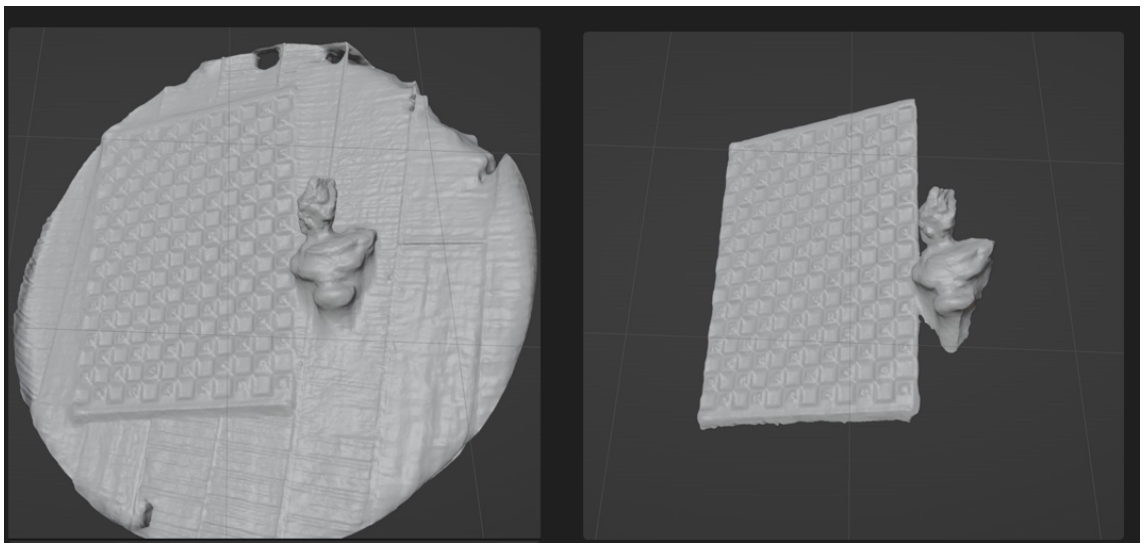

图 9. 基准与改进后的实验结果对比

6 总结与展望

我们利用了一些先验信息如稀疏点云和掩码进行监督模型的训练从而来优化我们重建的结果，但是这样的话会使得我们重建的质量对于稀疏点云和掩码的准确性更加依赖，这是我们的不足之处。未来我们也许不只是考虑对模型的损失函数进行优化，我们可以通过修改模型进行优化从而得到提升。

参考文献

[1] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[2] Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4287–4297, 2021.

[3] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023.

[4] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[5] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[6] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[7] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[8] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 501–518. Springer, 2016.

[9] Robust Multiview Stereopsis. Accurate, dense, and robust multiview stereopsis. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 32(8), 2010.

[10] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[11] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 803–814, 2023.

[12] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. Jan 2020.

[13] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020.