基于 TensoRF 和 3D Gaussian Splatting 实现物体重建与新视图合成

金潇睿

项目主页: https://github.com/Anderasderry/2025CV-final

一、 实验简介

本实验分别使用 TensoRF 框架和 3D Gaussian Splatting 实现了三维物体的重建与新视图合成。实验流程包括图像采集、相机参数估计、模型训练、渲染可视化等步骤。通过 COLMAP 恢复相机位姿，再利用各自框架进行稠密建模与图像生成。选用一个日常物体（咖啡杯）作为目标，围绕其拍摄短视频并进行数据处理与训练。

二、 TensoRF 训练与渲染

本实验在 github 仓库 TensoRF 提供的框架下进行模型训练与新视图合成。

1. 模型介绍：TensoRF

TensoRF（Tensorial Radiance Fields）是一种基于张量分解的 NeRF 加速方法。它将场景表示为一个高维的体素特征张量，并通过张量分解对该张量进行紧凑参数化。这样不仅大幅减少了存储需求，还显著加快了训练速度和渲染效率，同时保证了渲染质量。TensoRF的核心思路是用低秩张量重建的方式，将原本高维且冗余的voxel特征表达压缩成由向量和矩阵组成的因子，从而降低复杂度。相比原版 NeRF，TensoRF 的特点有：

方面	原版 NeRF	TensoRF
场景表示	使用连续函数（基于坐标的MLP网络）	使用稀疏多通道体素网格 + 张量分解
表达形式	通过全连接神经网络直接拟合颜色和密度	将3D体素特征张量分解为向量和矩阵，低秩张量表达
存储需求	MLP参数较少，但需大量计算	voxel grid本身很大， 但张量分解显著压缩
训练速度	训练时间长，需要数小时至数天	训练速度快，通常30分钟内完成， 速度提升100倍以上
渲染方式	MLP前向传播，较慢	基于查表和插值，渲染速度快
渲染质量	高质量但训练成本高	质量优于原版NeRF，且更快

2. 数据采集与预处理

- **图像获取**：围绕咖啡杯旋转拍摄了约 20 秒的视频，后用 `ffmpeg` 提取图像帧，得到 611 张图像
- **相机参数估计**：使用 COLMAP 工具恢复每张图像对应的相机姿态信息，具体流程如下：
  i. **特征提取（Feature Extraction）**
  提取图像的局部特征，并将其存储至数据库。

  ```
  colmap feature_extractor --database_path database.db --image_path my_cup
  ```

  ii. **顺序匹配（Sequential Matcher）**
  按图像顺序匹配相邻图像间的特征点，适用于视频帧或环拍图像序列。

  ```
  colmap sequential_matcher --database_path database.db
  ```

  iii. **稀疏重建（Mapper）**
  构建稀疏点云，并估计每张图像的内外参（相机位姿 + 内参矩阵），输出模型为 `sparse/0` 。

  ```
  colmap mapper --database_path database.db --image_path my_cup --output_path sparse
  ```

  iv. **模型转换（Model Converter）**
  将二进制格式转换为 TXT 格式，供后续训练使用。

  ```
  colmap model_converter --input_path sparse/0 --output_path sparse/0 --output_type TXT
  ```

- **格式转换与数据划分**
  使用 TensoRF 提供的 `colmap2nerf.py` 脚本将 COLMAP 模型转换为 NeRF 格式（`transforms.json`）：

  ```
  python colmap2nerf.py --colmap_dir sparse/0 --images my_cup --out_dir ./my_cup
  ```

  然后运行 `split_transforms.py`，划分出训练、验证和测试集，分别生成：
  - `transforms_train.json` （488张）
  - `transforms_val.json` （61张）
  - `transforms_test.json` （62张）

3. 实验参数设置

参数	取值
总迭代次数 `n_iters`	30000
批大小 `batch_size`	4096
初始体素数	2097156（128^3）
最终体素数	27000000（300^3）
上采样点	$[2000, 3000, 4000, 5500, 7000]$

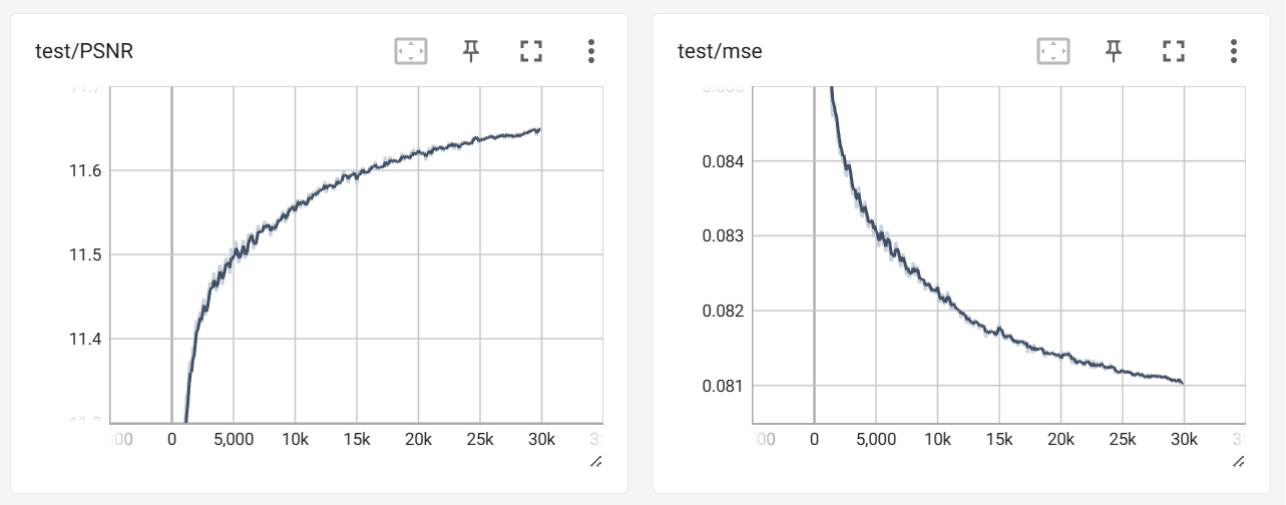
参数	取值
AlphaMask 更新点	$[2000, 4000]$
SH 维度	`n_lamb_sigma = [16,16,16]` , `n_lamb_sh = [48,48,48]`
着色器类型	`MLP_Fea`
激活函数	`softplus`
TV 正则化系数	density: 0.1, appearance: 0.01

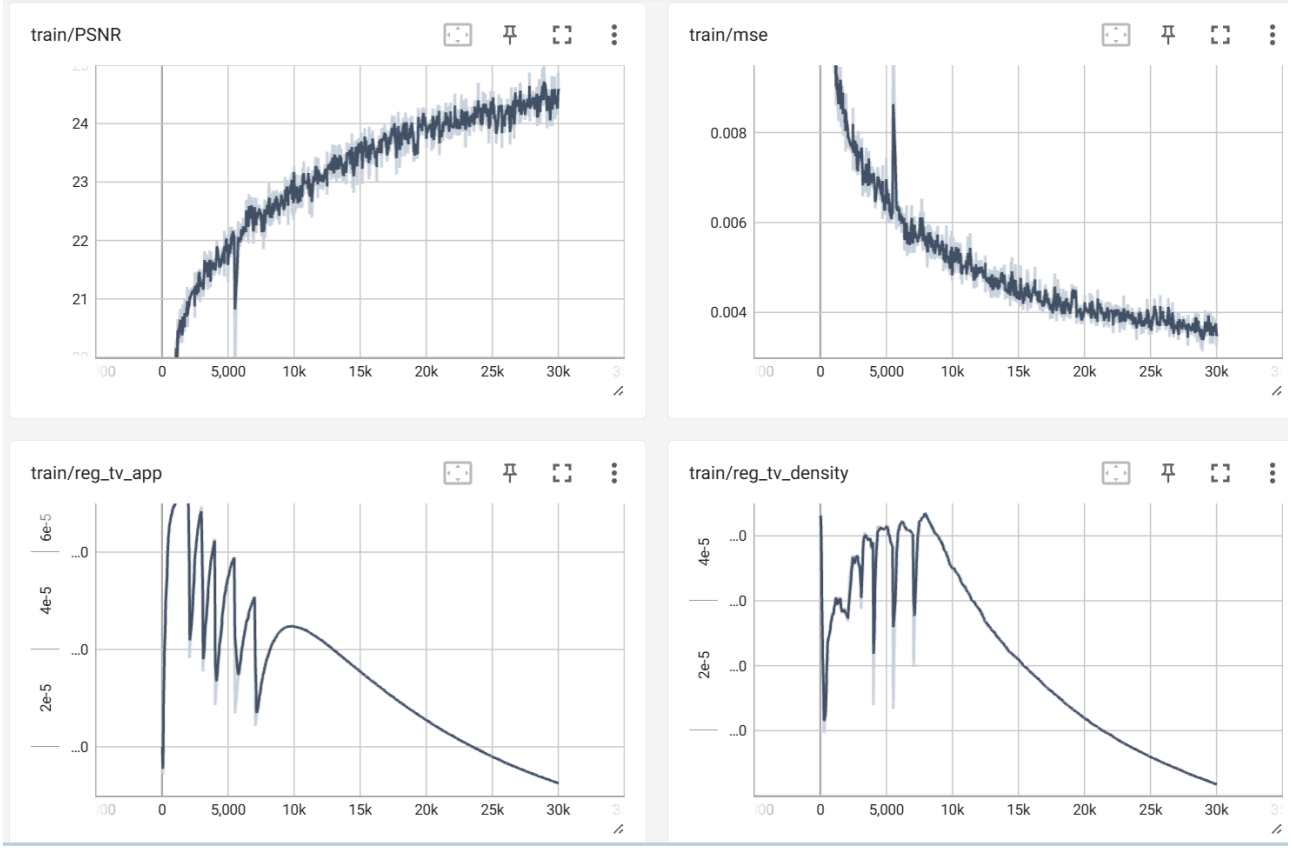
其中，为了防止模型过拟合并增强体素表达的空间连续性，TensoRF 在训练过程中对体素密度和颜色的空间梯度引入了 Total Variation（TV）正则化，分别作用于 density 和 appearance 张量。实验中设定的 TV 正则化系数为： `TV_density_weight = 0.1` , `TV_appearance_weight = 0.01` 。
在 `configs/my_cup.txt` 中配置好训练参数，输入以下指令开始训练：

```
python train.py --config configs/my_cup.txt
```

4. 训练过程可视化

训练过程中，每1000轮次在测试集上评估一次 PSNR 和 MSE 指标，在第 9999，19999，29999 轮次时保存模型参数和渲染的视频。训练日志保存在 `log` 目录下，使用 Tensorboard 查看训练过程中在训练集和测试集上的loss曲线，以及在测试集上的 PSNR 指标。





训练集与测试集的 MSE 损失均稳定下降，PSNR 曲线整体上升，说明模型重建质量持续优化。而 tv_density 和 tv_appearance 的正则化 loss 值在训练初期呈上升趋势，在中后期达到峰值后逐步下降。可能的原因是在初期阶段，模型尚未收敛，体素参数较为稀疏或不规则，随着训练进行，TensoRF 开始拟合更细致的颜色与密度分布，导致相邻体素之间的值差变大，从而 TV loss 增加；在后期模型收敛阶段，网络趋于平滑，TV 正则在优化器作用下成功抑制了局部高频波动，使 loss 值下降。

5. 训练结果

使用如下指令渲染视频：

```
python render.py --config configs/my_cup.txt --ckpt ($model_path$)
```

最终训练好的模型在测试图片上 PSNR 指标达到 11.528 dB，渲染好的视频已上传至百度网盘。

三、 3D Gaussian Splatting 训练与渲染

本实验在 github 仓库 gaussian-splatting 提供的框架下进行模型训练与新视图合成。

1. 模型介绍：3D Gaussian Splatting

3D Gaussian Splatting 是一种直接使用可渲染高斯点云对场景建模的方法。不同于 NeRF 和 TensoRF 需要通过体素或网络建模场景体积信息，它使用一组具有空间位置、协方差矩阵和颜色的高斯分布来表示场景中的光照与结构信息。

特点包括：

- 不需要 MLP 或 voxel grid；
- 直接优化空间中高斯点的位置、颜色和形状；
- 极高的渲染速度（支持实时渲染）；
- 高视觉保真度，PSNR、SSIM 等指标优异。

2. 数据采集与预处理

本部分使用与 TensoRF 相同的数据源，但处理方式略有不同。同样依托 COLMAP 生成稀疏点云与相机参数，但是使用 Gaussian Splatting 提供的 `convert.py` 工具将图片集转换为可用训练数据（`cameras.bin`，`images.bin`，`points3D.bin`）。将图片保存在 `data/images` 目录下，运行如下命令：

```
python convert.py -s data
```

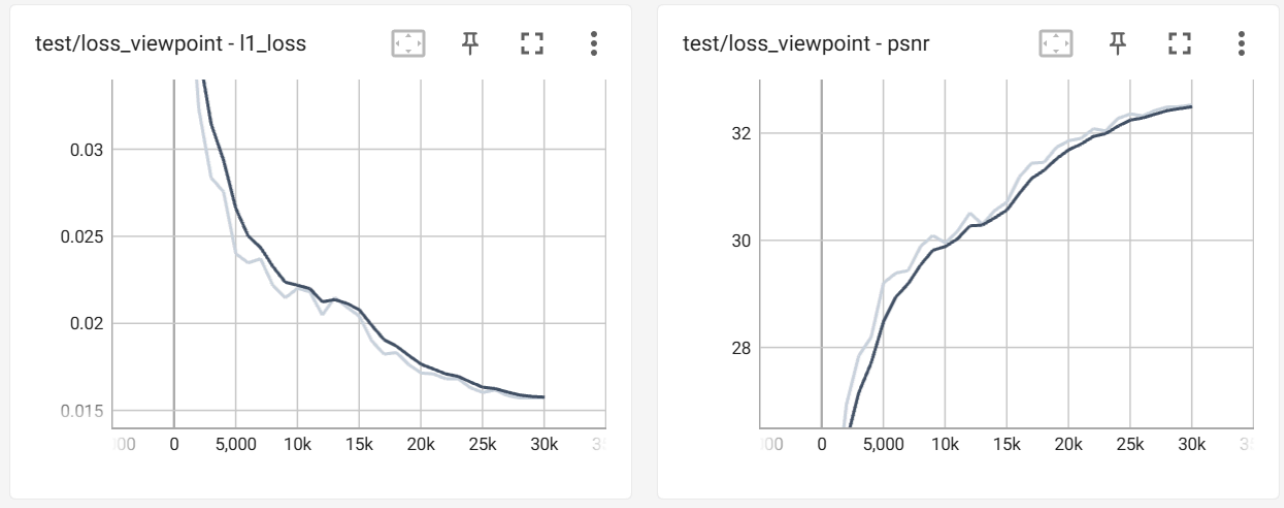
3. 实验参数设置

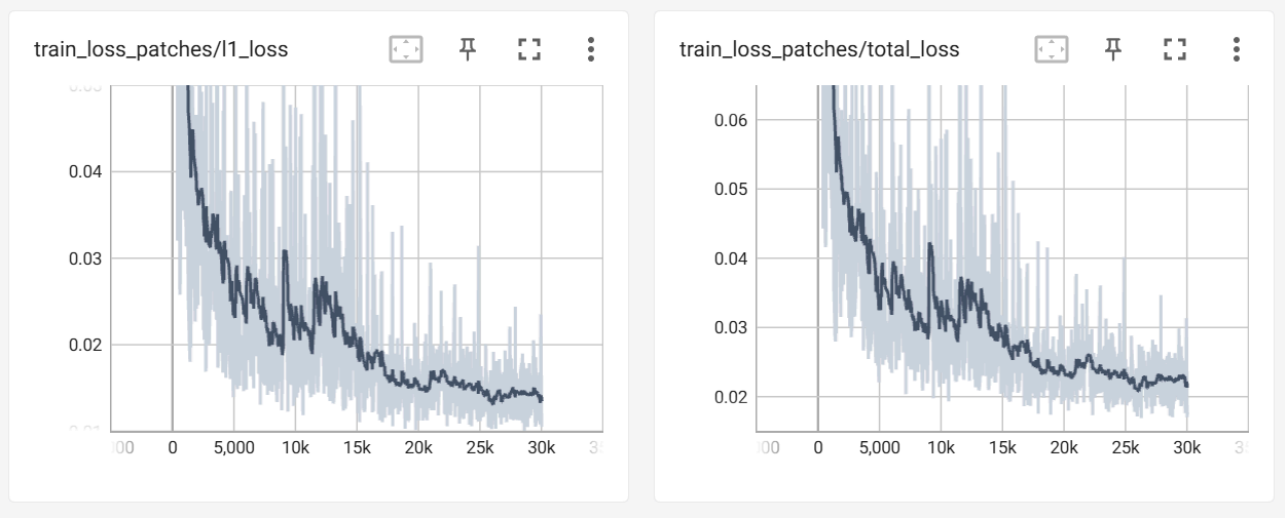
3D Gaussian Splatting 的训练基于其官方 PyTorch 实现进行，该实现采用可微高斯点云作为场景表示形式，并通过端到端优化其位置、协方差、颜色和方向性参数。训练流程中，每一步从训练集中随机选取一个相机视角，渲染对应图像并与 Ground Truth 进行 L1 + SSIM 联合损失优化。每隔固定迭代周期执行稠密化操作（densification），通过梯度阈值和尺寸限制生成新的点，提升渲染精度。实验中默认设置训练总步数为 30,000，测试和保存分别在第 7000 步和第 30000 步进行。同时，系统通过 TensorBoard 自动记录训练和测试集的损失及 PSNR 指标，并输出渲染图像结果。该方法无需传统神经网络结构，训练与渲染效率远高于 NeRF 系列方法。

训练命令如下：

```
python train.py -s data
```

4. 训练过程可视化





训练集与测试集的 MSE 损失均稳定下降，PSNR 曲线整体上升，说明模型重建质量持续优化。

5. 训练结果

训练输出在 `output` 目录下，其中包含训练日志、模型权重、渲染结果等。执行如下指令：

```
python render.py --model_path output/point_cloud --iteration 30000
```

会在 `output/point_cloud/train/ours_30000/renders/` 下生成渲染图像。

最终训练好的模型在测试图片上 PSNR 指标达到 32.475 dB，渲染好的视频已上传至百度网盘。另外，还使用 SIBR 对点云文件（.ply）进行了渲染，可以自由切换角度观察渲染好的物体，操作录屏也上传至网盘。

四、总结与对比

方法	训练时间	渲染速度	表达形式	是否支持实时渲染
原版 NeRF（理论）	数小时~天	慢	MLP（隐式）	否
TensoRF	30分钟左右	快	张量体素网格	可
3D Gaussian Splatting	十几分钟	非常快	显式可微高斯点云	是

根据实验结果可得，TensoRF 在表达效率和模型压缩上具有独特优势，而 Gaussian Splatting 则在视觉保真和实时渲染方面表现出色。

附录

- GitHub 仓库：https://github.com/Anderasderry/2025CV-final
- 模型权重和视频下载地址：https://pan.baidu.com/s/1WGyPdzT63IBT-AZIzo9ywQ，提取码：qypb