



浙江大学

ZHEJIANG UNIVERSITY

浙江大学软件学院优秀大学生夏令营
实验报告

姓 名 _____ 张宏远 _____

选 题 _____ 任务 1+子任务 3 _____

目录

一、子任务 3	3
3.1 Street Gaussians 论文复现	3
3.1.1 数据处理	3
3.1.2 复现步骤	4
3.1.3 实验结果	7
3.1.4 改进意见	10
3.2 3D Gaussian 论文复现	10
3.2.1 数据处理	10
3.2.2 复现步骤	11
3.2.3 实验结果	12
3.3 HUGS 论文复现	14
3.3.1 数据处理	14
3.3.2 复现步骤	14
3.3.3 实验结果	15
二、附录	16
三、引用	17

一、子任务 3

任务要求：选择一个在子任务 1 中提及的数据集，并进行部分数据的下载。随后，采用子任务 2 中阐述的方法，下载相关开源代码并进行训练复现，对重建结果进行精度评估（PSNR, SSIM, LPIPS 等）。针对在重建过程中发现的不足之处，提出具体的改进意见和建议。最后，撰写一份详细的报告，涵盖数据处理过程、复现步骤、实验结果以及改进意见等内容。

完成概要：在这个部分，我复现了三篇论文的代码。对于第一篇论文 Street Gaussians 的复现，我是按照子任务 3 的任务要求进行的。其余两篇论文的复现是在要求的基础上又额外去做的工作，其中在 3D Gaussian 这篇论文的复现中，我选取了三个场景进行点云生成。而对于 HUGS 这篇论文的复现，由于其代码并未完全开源，所以我只复现了一部分代码，并进行了精度评估。

3.1 Street Gaussians 论文复现

在这一部分，我对 *Street Gaussians for Modeling Dynamic Urban Scenes* [1]这一论文的代码进行了复现，这里我选择了 Waymo 数据集进行训练。首先，搭建了必要的环境，确保能够顺利下载和处理数据。接下来，我对数据进行了预处理，包括数据清洗和格式转换，以便适应后续模型训练需求。在复现代码部分，我参考了已有的研究成果，严格按照既定步骤进行实现，确保代码的可复现性和可靠性。最后，通过精度评估对模型的性能进行了验证，并利用点云和视频截取数据进行了可视化展示。代码在 AutoDL 平台上进行运行，GPU 为 RTX 4090。

3.1.1 数据处理

在开始复现论文之前，首先要配置 python 的运行环境，如图 1 所示。

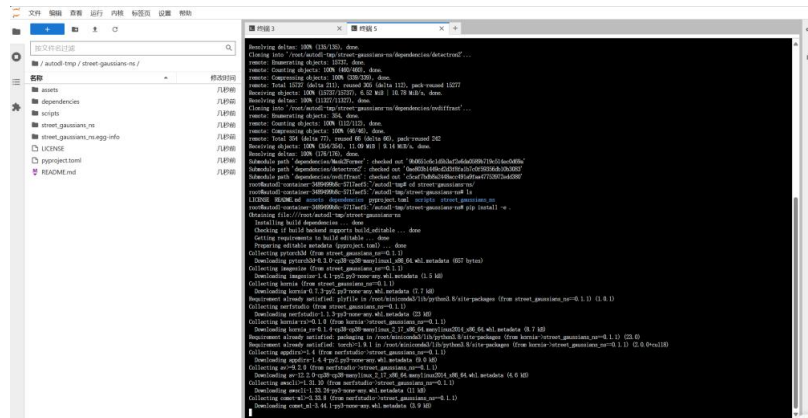


图 1 Street Gaussians 代码环境配置

在数据处理部分，论文使用了 waymo perception_v1.4.0 数据集。这里我们需要对数据集进行预处理，使用如下命令：

`bash scripts/shells/points_cloud_generate.sh /root/autodl-tmp/street-gaussians-ns/data`
生成数据集中点云相关文件，如图 2 所示，生成的点云文件包含在图 3 中。

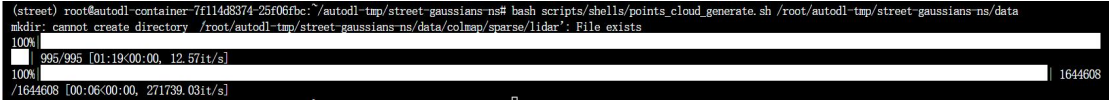


图 2 数据集处理过程

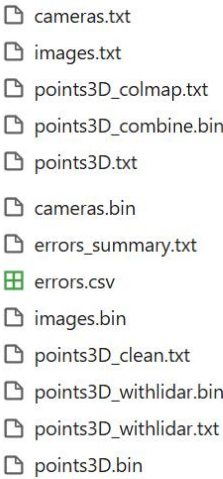


图 3 点云文件

3.1.2 复现步骤

在复现步骤首先先对数据集进行训练，训练命令如下：
`bash scripts/shells/train.sh "/root/autodl-tmp/street-gaussians-ns/data" 0`
训练过程如图 5 到图 8。

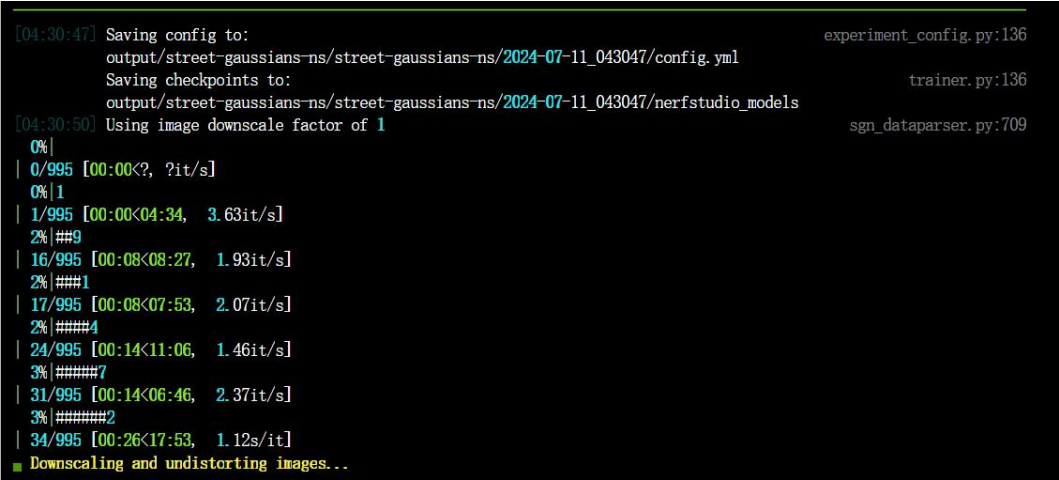


图 4 训练 1：处理的数据 1

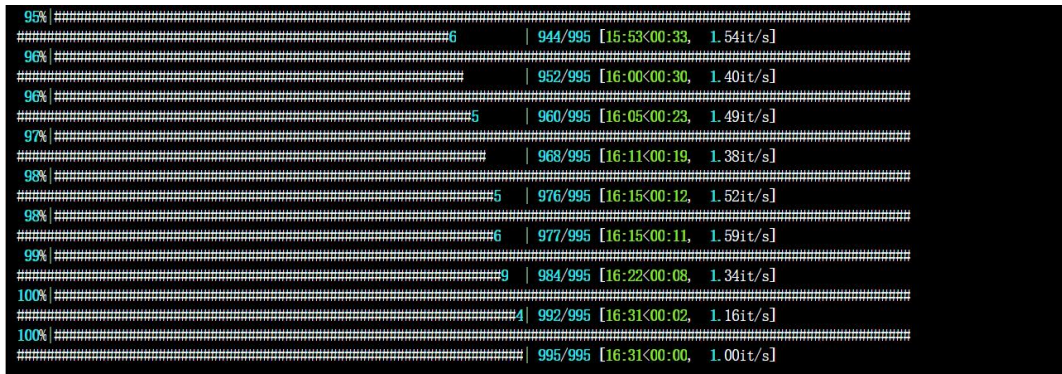


图 5 训练 2：处理的数据 2

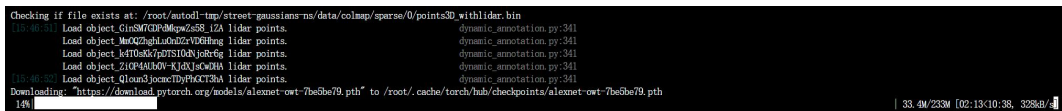


图 6 训练 3：下载预训练模型

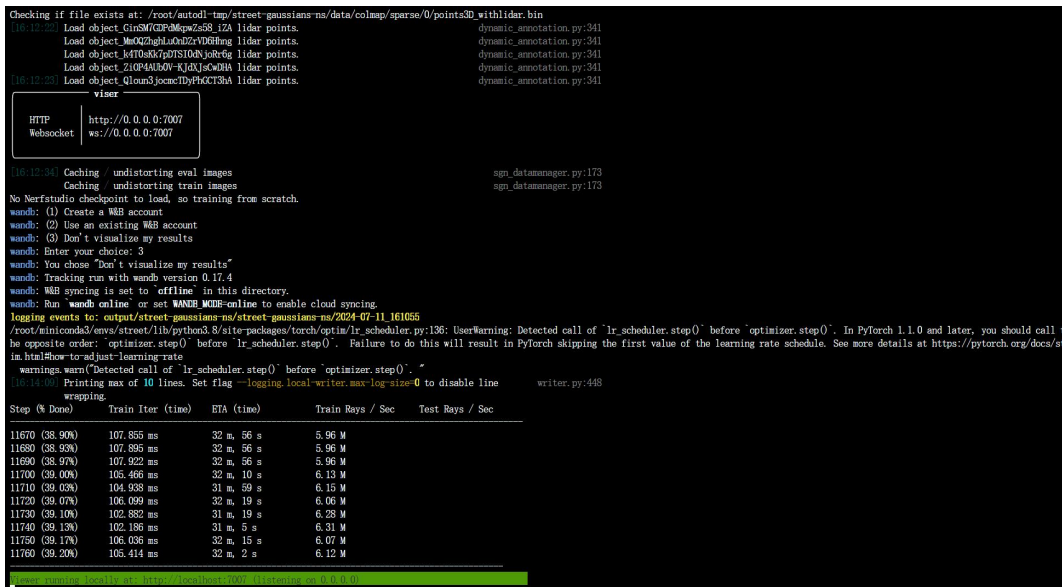


图 7 训练 4：训练进度展示

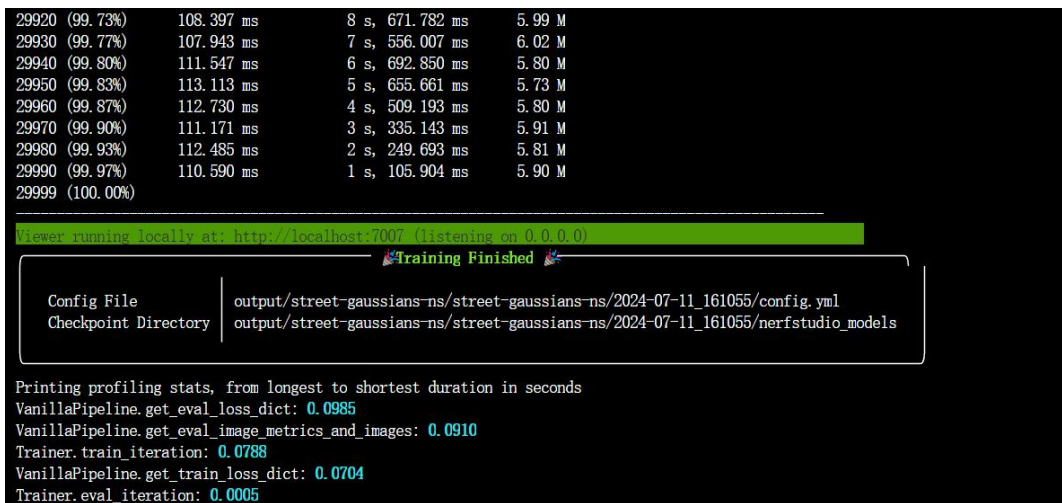


图 8 训练 5：性能分析展示

在运行完训练代码后，还需对模型的性能进行评估并且对生成的数据进行渲染，首先这里运行对模型的性能进行评估的脚本如下所示

```
bash scripts/shells/eval.sh
```

```
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/config.yml 0
```

代码的运行结果如图 9 所示。

```
99% #####
#####3 | 984/995 [00:24<00:00, 46.22it/s]
100% #####
#####5 | 992/995 [00:24<00:00, 46.09it/s]
100% #####
##### | 995/995 [00:24<00:00, 41.05it/s]

Checking if file exists at: /root/autodl-tmp/street-gaussians-ns/data/colmap/sparse/0/points3D_withlidar.bin
[14:32:55] Load object_GinSM7GDPdMkpWZs58_iZA lidar points. dynamic_annotation.py:341
Load object_MmOQZhghLuOnDZrVD6Hhng lidar points. dynamic_annotation.py:341
Load object_k4T0sKk7pDTsIOdNjoRr6g lidar points. dynamic_annotation.py:341
Load object_ZiOP4AUbOV-KJdXjsCwDHA lidar points. dynamic_annotation.py:341
[14:32:56] Load object_Qloun3jocmcTDyPhGCT3hA lidar points. dynamic_annotation.py:341
Loading latest checkpoint from load_dir
Done loading checkpoint from
output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/nerfstudio_models/step-000029999.ckpt
[14:33:06] Caching / undistorting eval images sgn_datamanager.py:173
Saved results to:
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/eval_output.json
{ 'experiment_name': 'street-gaussians-ns', 'method_name': 'street-gaussians-ns', 'checkpoint':
'output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/nerfstudio_models/step-000029999.ckpt', 'results':
{ 'psnr': 30.36919593811035, 'psnr_std': 1.551006555557251, 'ssim': 0.9199024438858032, 'ssim_std': 0.01675284281373024,
'lpips': 0.1725744754076004, 'lpips_std': 0.02610611356794834, 'num_rays_per_sec': 24221072.0, 'num_rays_per_sec_std':
9689194.0, 'fps': 9.855579376220703, 'fps_std': 3.9425430297851562}}
```

图 9 评估代码运行过程及结果

然后还需要对训练后的数据进行进一步的渲染，其渲染所用脚本如下所示：

```
bash scripts/shells/render.sh
```

```
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/config.yml 0
```

最终将渲染的结果报错到了输出目录中，如图 10 所示。

```
Checking if file exists at: /root/autodl-tmp/street-gaussians-ns/data/colmap/sparse/0/points3D_withlidar.bin
[19:38:25] Load object_GinSM7GDPdMkpWZs58_iZA lidar points. dynamic_annotation.py:341
Load object_MmOQZhghLuOnDZrVD6Hhng lidar points. dynamic_annotation.py:341
Load object_k4T0sKk7pDTsIOdNjoRr6g lidar points. dynamic_annotation.py:341
Load object_ZiOP4AUbOV-KJdXjsCwDHA lidar points. dynamic_annotation.py:341
[19:38:26] Load object_Qloun3jocmcTDyPhGCT3hA lidar points. dynamic_annotation.py:341
Rendering split all 90/90(100.0%) 2.60 fps 0:00:00 0:00:37
Render on split all Complete
Outputs all | /root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_16...
```

图 10 输出点云的过程及结果

在进行完上述步骤后，可以额外进行一步将将 nerfstudio 的模型导出为 3DGS 格式的 ply 文件方便在其他环境下对结果进行预览，运行如下指令：

```
bash scripts/shells/export.sh
```

```
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/
```

最终会生成场景和车辆的点云 ply 文件，如图 11 所示。我们可以使用 CloudCompare、虚幻引擎或者其他可以查看点云文件的软件进行查看。

```
Checking if file exists at: /root/autodl-tmp/street-gaussians-ns/data/colmap/sparse/0/points3D_withlidar.bin
[17:41:04] Load object_GinSM7GDPdMkpWZs58_i2A lidar points. dynamic_annotation.py:341
Load object_MmOQZhghLuOnDZrVD6Hhng lidar points. dynamic_annotation.py:341
Load object_k4T0sKk7pDTSIOdNjoRr6g lidar points. dynamic_annotation.py:341
Load object_ZiOP4AubOV-KJdXJsCwDHA lidar points. dynamic_annotation.py:341
[17:41:05] Load object_Qloun3jocmcTdyPhCCT3hA lidar points. dynamic_annotation.py:341
Loading latest checkpoint from load_dir
Done loading checkpoint from
output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/nerfstudio_models/step-000029999.ckpt
Saved PLY to
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/point_cloud_backgr
ound.ply
Saved PLY to
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/point_cloud_object
_GinSM7GDPdMkpWZs58_i2A.ply
Saved PLY to
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/point_cloud_object
_MmOQZhghLuOnDZrVD6Hhng.ply
Saved PLY to
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/point_cloud_object
_k4T0sKk7pDTSIOdNjoRr6g.ply
Saved PLY to
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/point_cloud_object
_ZiOP4AubOV-KJdXJsCwDHA.ply
Saved PLY to
/root/autodl-tmp/street-gaussians-ns/output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/point_cloud_object
_Qloun3jocmcTdyPhCCT3hA.ply
```

图 11 生成点云运行结果

3.1.3 实验结果

项目最终的输出结果包含很多中不同的类型，接了来我将一一展示，其中模型
的最终性能指标的 json 格式文件如图 12 所示。

```
1 eval_output.json > ...
2 {
3   "experiment_name": "street-gaussians-ns",
4   "method_name": "street-gaussians-ns",
5   "checkpoint": "output/street-gaussians-ns/street-gaussians-ns/2024-07-11_161055/nerfstudio_models/step-000029999.ckpt",
6   "results": {
7     "psnr": 30.36919593811035,
8     "psnr_std": 1.551006555557251,
9     "ssim": 0.9199024438858032,
10    "ssim_std": 0.01675284281373024,
11    "lpips": 0.1725744754076004,
12    "lpips_std": 0.02610611356794834,
13    "num_rays_per_sec": 23895206.0,
14    "num_rays_per_sec_std": 9836111.0,
15    "fps": 9.722984313964844,
16    "fps_std": 4.002324104309082
17  }
```

图 12 性能指标

项目在渲染的步骤中会生成多种渲染得到的视频结果，其中包含了最终渲染
的 RGB 视频，此外还有与深度、天空、背景、车辆渲染相关的视频。展示内容
如图 13、图 14、图 15 所示。

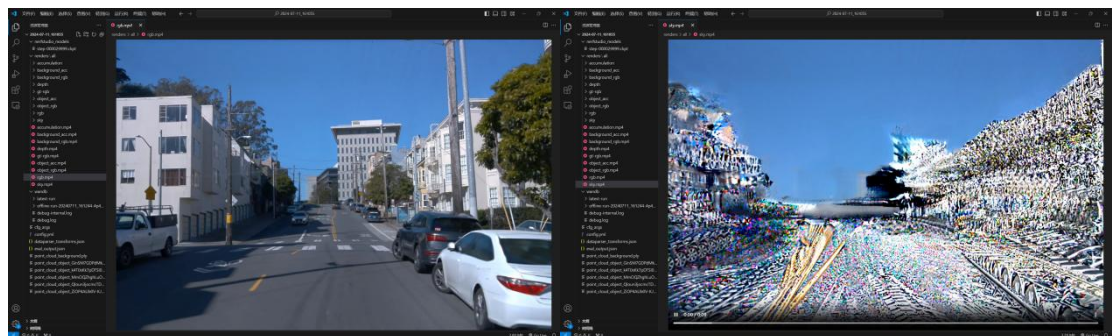


图 13 渲染视频 1

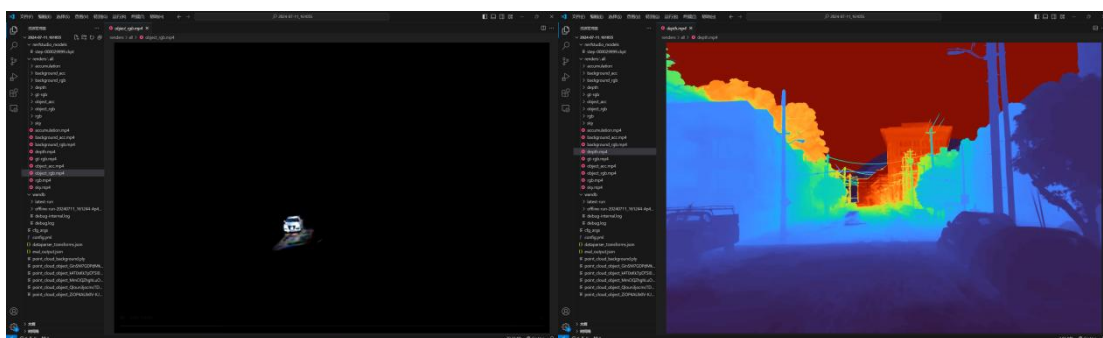


图 14 渲染视频 2

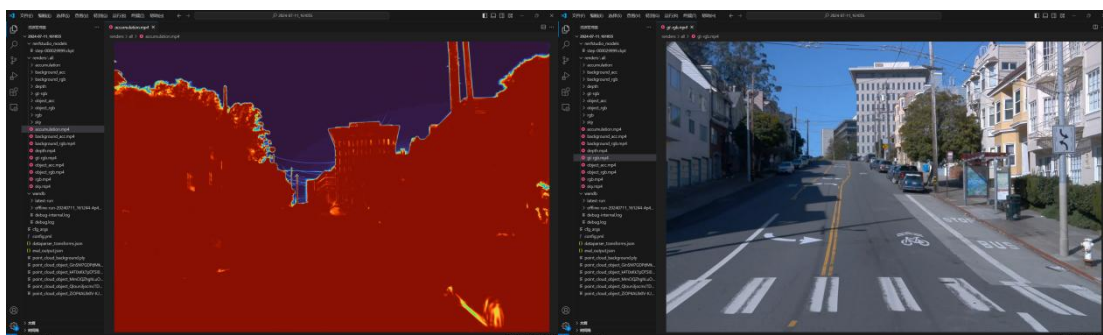


图 15 渲染视频 3

此外在导出步骤结束后会将 nerfstudio 的模型导出为 3DGS 格式的 ply 这里我使用 CloudCompare 软件和利用虚幻引擎加载点云文件进行加载生成的点云文件。这里效果如图 16、图 17、图 18 和图 19 所示，其中图 18、图 19 是将汽车和街景通过插件加载至虚幻引擎的效果图，配合上天空场景仿真效果优异。

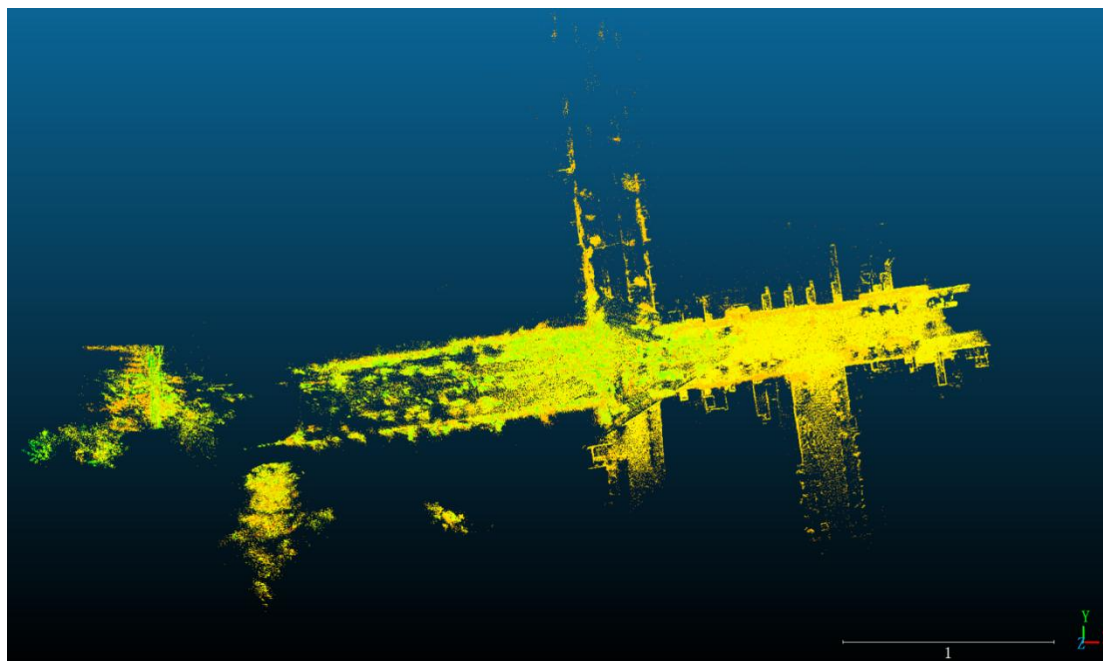


图 16 Street Gaussian 实验结果 1

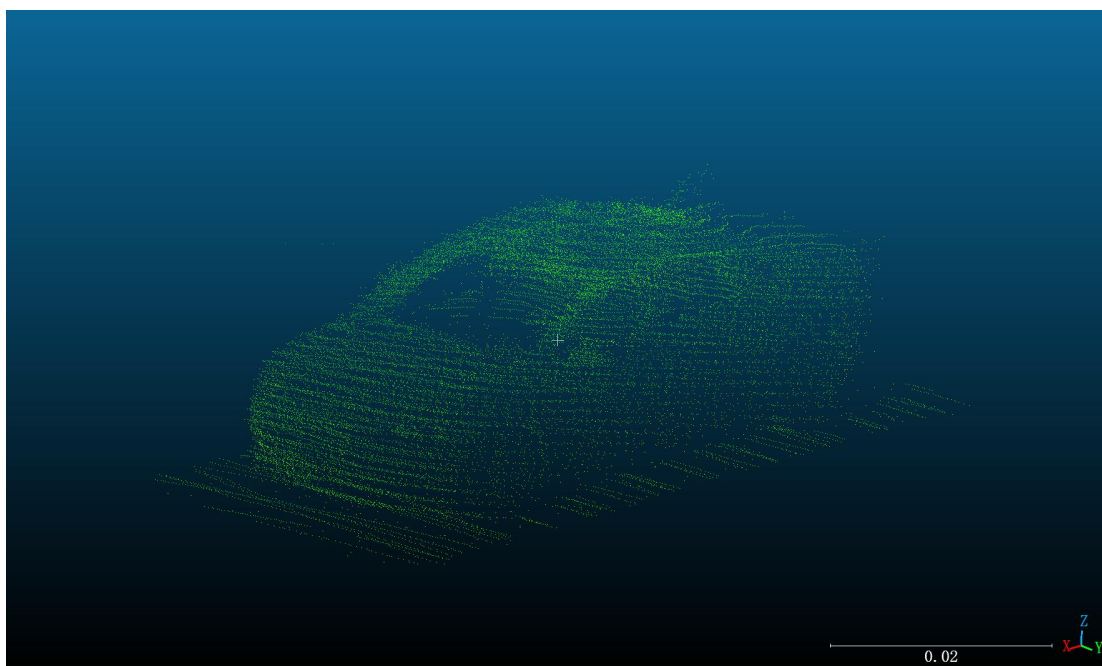


图 17 Street Gaussian 实验结果 2



图 18 Street Gaussian 实验结果 3



图 19 Street Gaussian 实验结果 4

3.1.4 改进意见

对于论文本身的改进意见我在子任务 2 中已经给出了很详细的阐述，这里我仅仅对于项目代码层面提出改进意见。首先项目本身效果已经非常出众了，但是我在环境配置中还是发现项目中某些环境可能存在不兼容的问题，比如项目中需要用到 1.0 版本的 nerfstudio，而其最高支持的 gsplat 版本无法驱动项目的运行，需要下载 0.1.9 版本的 gsplat，而这会导致和 nerfstudio 版本冲突。项目中需要用到的原始数据集为 waymo perception_v1.4.0 目前无法下载，这边只能通过预处理后的数据集然后再进行生成点云文件的处理。

3.2 3D Gaussian 论文复现

这里我对 3D Gaussian Splatting for Real-Time Radiance Field Rendering [2]这一论文的代码进行了复现。训练数据来自三个场景：我的实验室、宿舍的钢铁侠和校园空旷场所。数据处理和点云展示部分我在 Windows 系统下进行，而代码的复现部分则选择在 AutoDL 平台上进行，使用的 GPU 是 RTX 4090。

3.2.1 数据处理

在本次论文复现中，我没有选择子任务 1 中介绍的数据集，而是通过自己录制视频的方法来获取训练数据。为此，这里编写了一个视频处理程序（代码见附录代码 6），用于对视频进行切分帧数，并生成必要的数 据，最终生成的数据结构如图 20 所示。

/ autodl-tmp / data1 /

名称	修改时间
distorted	3 小时前
images	3 小时前
input	3 小时前
sparse	3 小时前
stereo	3 小时前
run-colmap...	4 小时前
run-colmap...	4 小时前

图 20 程序处理后的数据结构

3.2.2 复现步骤

在 AutoDL 平台上实现论文代码的复现，首先先使用该模型训练上述处理后的数据集，使用指令：`python train.py -s "/root/autodl-tmp/data1" --eval`。--eval 代表着指示脚本执行评估任务，运行的过程如图 21 所示。

```
(gaussian_splatting) root@autodl-container-8b8a4382f2-e87f06d2:/autodl-tmp/gaussian-splatting# python train.py -s "/root/autodl-tmp/ironman" --eval
Optimizing
Output folder: ./output/7907ed5c-f [11/07 03:04:43]
Tensorboard not available: not logging progress [11/07 03:04:43]
Reading camera 221/221 [11/07 03:04:45]
Loading Training Cameras [11/07 03:04:43]
[ INFO ] Encountered quite large input images (>1.6K pixels width), rescaling to 1.6K.
If this is not desired, please explicitly specify '--resolution/-r' as 1 [11/07 03:04:43]
Loading Test Cameras [11/07 03:04:58]
Number of points at initialization: 10710 [11/07 03:04:59]
Training progress: 28% | 7000/30000 [02:17:07:48, 49.13it/s, Loss=0.0215560]
[ITER 7000] Evaluating test: L1 0.030101564147376887 PSNR 29.07133913040161 [11/07 03:07:17]
[ITER 7000] Evaluating train: L1 0.0141653785482049 PSNR 33.9940803278321 [11/07 03:07:17]
[ITER 7000] Saving Gaussians [11/07 03:07:17]
Training progress: 100% | 30000/30000 [10:41:00:00, 46.75it/s, Loss=0.0176924]
[ITER 30000] Evaluating test: L1 0.025766115469323866 PSNR 31.001957825251985 [11/07 03:15:41]
[ITER 30000] Evaluating train: L1 0.009017328545451165 PSNR 37.641427993774414 [11/07 03:15:41]
[ITER 30000] Saving Gaussians [11/07 03:15:41]
Training complete. [11/07 03:15:43]
```

图 21 3D Gaussian 训练过程

运行脚本：`python render.py -m /root/autodl-tmp/gaussian-splatting/output/data1` 进行渲染步骤，渲染的过程如图 22 所示。

```
(gaussian_splatting) root@autodl-container-8b8a4382f2-e87f06d2:/autodl-tmp/gaussian-splatting# python render.py -m "/root/autodl-tmp/gaussian-splatting/output/iron"
Looking for config file in /root/autodl-tmp/gaussian-splatting/output/iron/cfg_args
Config file found: /root/autodl-tmp/gaussian-splatting/output/iron/cfg_args
Rendering /root/autodl-tmp/gaussian-splatting/output/iron
Loading trained model at iteration 30000 [11/07 03:16:10]
Reading camera 221/221 [11/07 03:16:11]
Loading Training Cameras [11/07 03:16:11]
[ INFO ] Encountered quite large input images (>1.6K pixels width), rescaling to 1.6K.
If this is not desired, please explicitly specify '--resolution/-r' as 1 [11/07 03:16:11]
Loading Test Cameras [11/07 03:16:25]
Rendering progress: 100% | 193/193 [02:15:00:00, 1.43it/s]
Rendering progress: 100% | 28/28 [00:19:00:00, 1.42it/s]
```

图 22 3D Gaussian 渲染过程

运行脚本：`python metrics.py -m /root/autodl-tmp/gaussian-splatting/output/data1` 进行模型效果评估步骤，评估的过程如图 23 所示，其具体的评估指标如图 24 所示。

```
(gaussian_splatting) root@autodl-container-8b8a4382f2-e87f06d2:/autodl-tmp/gaussian-splatting# python metrics.py -m "/root/autodl-tmp/gaussian-splatting/output/iron"
Scene: /root/autodl-tmp/gaussian-splatting/output/iron
Method: ours_30000
Metric evaluation progress: 0% | 0/28 [00:00:00, ?it/s]
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to /root/.cache/torch/hub/checkpoints/vgg16-397923af.pth | 528M/528M [35:11:00:00, 262kB/s]
100% | 528M/528M [35:11:00:00, 262kB/s]
Downloading: "https://raw.githubusercontent.com/richzhang/PerceptualSimilarity/master/lips/weights/v0.1/vgg.pth" to /root/.cache/torch/hub/checkpoints/vgg.pth | 7.12k/7.12k [00:00:00:00, 2.78kB/s]
100% | 28/28 [35:58:00:00, 77.10s/it]
Metric evaluation progress: 100%
SSIM : 0.9599052
PSNR : 30.9715137
LPIPS: 0.2436039
```

图 23 3D Gaussian 评估过程

```
SSIM :    0.9599052
PSNR :    30.9715137
LPIPS:    0.2436039
```

图 24 3D Gaussian 评估结果

3.2.3 实验结果

在 3D Gaussian 建模中，我选择了三种不同的场景来进行点云建模：一个工位场景、校园开阔场景和宿舍钢铁侠场景。效果图如图 25 到图 30。

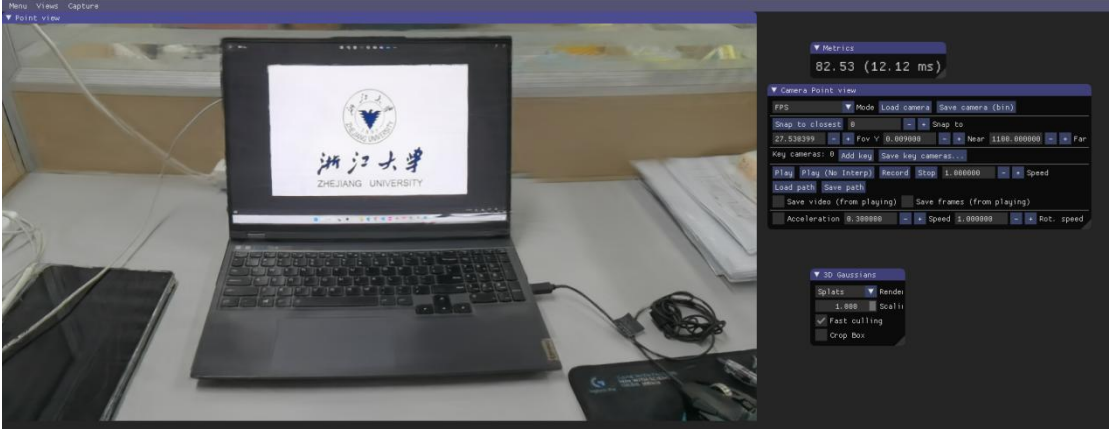


图 25 3D Gaussian 场景一

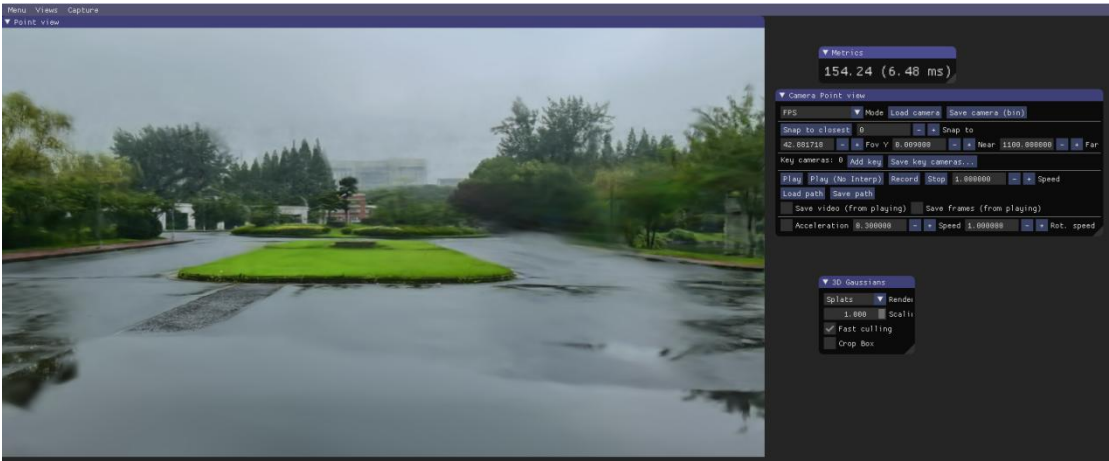


图 26 3D Gaussian 场景二 角度一

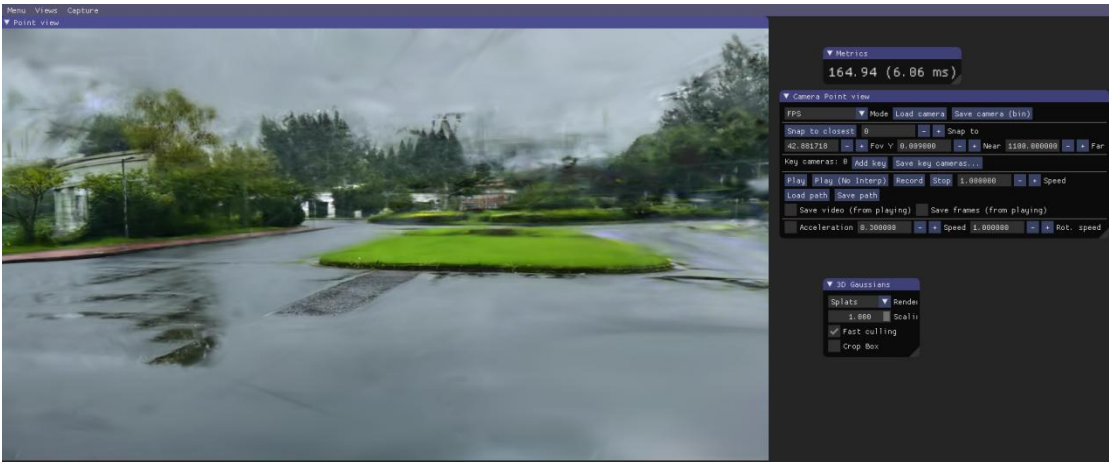


图 27 3D Gaussian 场景二 角度二

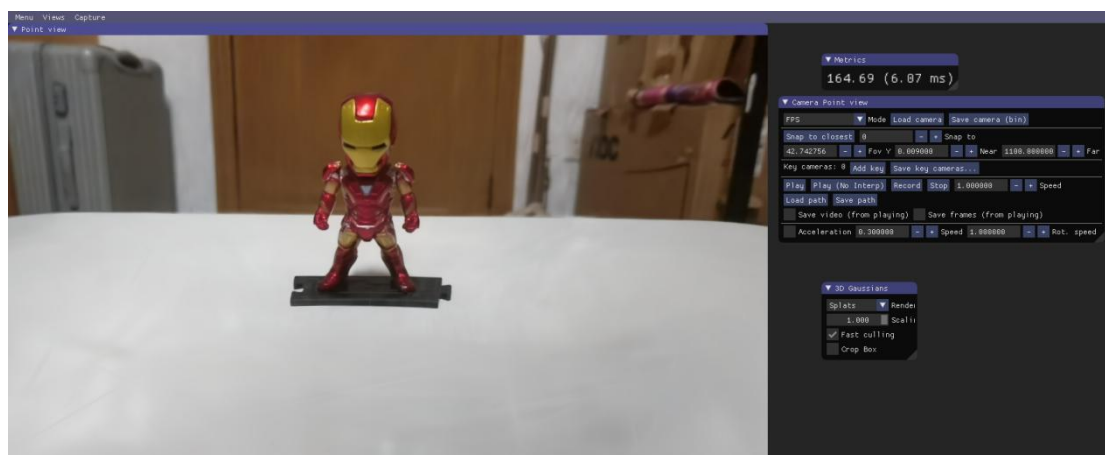


图 28 3D Gaussian 场景三 角度一

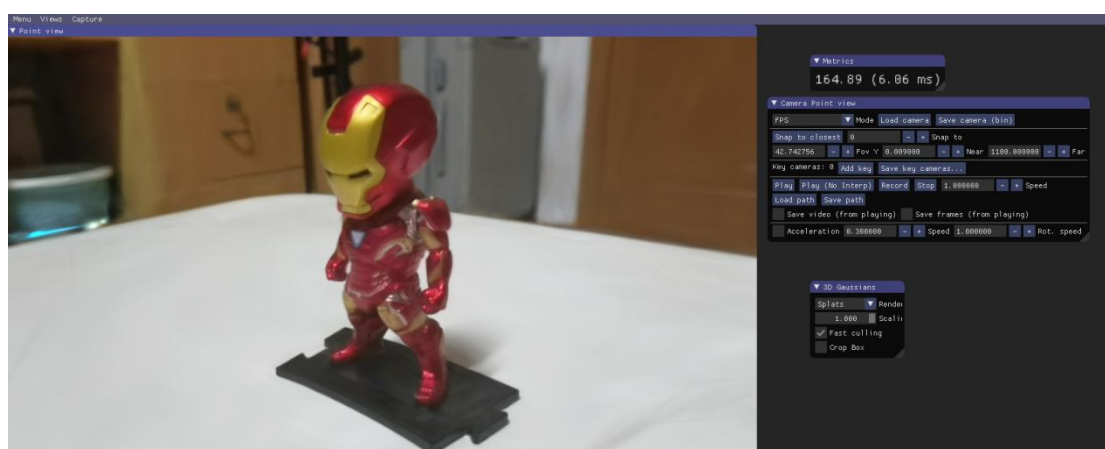


图 29 3D Gaussian 场景三 角度二



图 30 3D Gaussian 场景三 虚幻引擎展示

3.3 HUGS 论文复现

在这一部分是我对 *HUGS: Holistic Urban 3D Scene Understanding via Gaussian Splatting* [3]这篇论文的代码进行了复现，其论文的地址为(CVPR)。

由于时间原因，我没有仔细阅读这篇论文，只是略读了其创新点。论文的创新点在于利用预测的 2D 语义标签、光流和 3D 轨迹，结合 3D Gaussian Splatting 技术，从 RGB 图像中实现对城市场景的全面理解，无需额外的 LiDAR 扫描或手动标注的 3D 边界框。具体来说，论文通过将场景分解为静态区域和刚性移动的动态对象，每个动态对象使用 3D Gaussian 模型在其规范空间中表示，然后根据受单车模型约束的变换将其转换到世界坐标系。每个 3D Gaussian 模型包含关于外观和语义的信息，光流则通过计算高斯中心的运动获得，从而实现 RGB 图像、语义地图和光流的渲染。预测的 2D 语义标签通过使用 RGB 图像和噪声标注的 2D 语义标签进行训练得到。以下是我对论文的程序进行复现的内容。这里由于该项目环境较为简单，我选择在 Windows 系统进行运行，GPU 为 RTX3060。

3.3.1 数据处理

这篇论文的代码支持两个 KITTI 序列、三个 KITTI-360 序列和一个 Waymo 序列，我已经全部下载下来，如图 31 所示。由于论文支持直接使用数据集的原始格式，所以不需要对数据进行额外处理，直接解压即可。







 KITTI_02_140_f85_qd3dt.zip	2024/7/10 15:01	WinRAR ZIP 压缩文件	649,114 KB
 KITTI_06_65_f56_qd3dt.zip	2024/7/10 15:01	WinRAR ZIP 压缩文件	687,249 KB
 KITTI360_0000_5200_5266.zip	2024/7/10 15:07	WinRAR ZIP 压缩文件	1,615,971 KB
 KITTI360_0000_5335_5415.zip	2024/7/10 15:06	WinRAR ZIP 压缩文件	1,498,939 KB
 KITTI360_0000_5954_6034.zip	2024/7/10 15:07	WinRAR ZIP 压缩文件	1,751,918 KB
 Waymo_176124_0_102.zip	2024/7/10 15:08	WinRAR ZIP 压缩文件	2,942,473 KB

图 31 HUGS 所需数据集

3.3.2 复现步骤

在配置好 python 的运行环境后，运行如下指令：
`python render.py -m E:\A_Python\hugs\data\KITTI_06_65_f56_qd3dt --data_type kitti --iteration 30000 --affine`
对数据集进行渲染，其运行的过程如图 32 所示。



图 32 HUGS 渲染过程

运行指令 `python metrics.py -m E:A_Python\hugs\data\KITTI_06_65_f56_qd3dt` 进行评估模型的性能指标，如图 33 所示。

```
(hugs) PS E:\A_Python\hugs> python metrics.py -m E:\A_Python\hugs\data\KITTI_06_65_f56_qd3dt

Scene: E:\A_Python\hugs\data\KITTI_06_65_f56_qd3dt
Method: ours_30000
Metric evaluation progress: 0%| 0/25 [00:00<?, ?it/s]D
:Anaconda\envs\hugs\lib\site-packages\torchvision\models_utils.py:135: UserWarning: Using 'weights' as positional parameter(s) is deprecated since 0.13 and may be removed in the future. Please use keyword parameter(s) instead.
  warnings.warn(
:Anaconda\envs\hugs\lib\site-packages\torchvision\models_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing 'weights=AlexNet_Weights.IMAGENET1K_V1'. You can also use 'weights=AlexNet_Weights.DEFAULT' to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to C:\Users\45451\cache\torch\hub\checkpoints\alexnet-owt-7be5be79.pth
100%| 233M/233M [00:17<00:00, 13.8MB/s]
Downloading: "https://raw.githubusercontent.com/richzhang/perceptualSimilarity/master/loips/weights/v0.1/alex.pth" to C:\Users\45451\cache\torch\hub\checkpoints\alex.pth
100%| 232M/233M [00:17<00:00, 14.1MB/s]
5.87k/5.87k [00:00<00:00, 6.83MB/s]
Metric evaluation progress: 100%| 25/25 [00:30<00:00, 1.23s/it]
SSIM : 0.9131930
PSNR : 26.6772747
LPIPS: 0.0325859

Method: ours_30000
Metric evaluation progress: 100%| 62/62 [00:22<00:00, 2.73it/s]
SSIM : 0.9207560
PSNR : 27.7937775
LPIPS: 0.0285865
```

图 33 HUGS 评估模型过程

3.3.3 实验结果

评估模型的性能指标结果如图 34 所示。由于该论文只公开了 HUGS 的渲染评估代码，而训练代码并未开源，因此复现过程已经展示完毕。

```
Metric evaluation progress: 100%|
SSIM : 0.9131930
PSNR : 26.6772747
LPIPS: 0.0325859

Method: ours_30000
Metric evaluation progress: 100%|
SSIM : 0.9207560
PSNR : 27.7937775
LPIPS: 0.0285865
```

图 34 HUGS 性能指标

二、附录

以下代码均在 GitHub 上开源，开源地址如下。

(<https://github.com/Metastarx/ZJU>)

```
1. import os
2. import subprocess
3.
4. # 视频绝对路径
5. video_path = r"E:\A_Python\gaussian-splatting\data\data1\data1.mp4"
6. # 切分帧数，每秒多少帧
7. fps = 4
8.
9. # 获取当前工作路径
10. current_path = os.getcwd()
11. # 上一级文件夹所在路径
12. folder_path = os.path.dirname(video_path)
13. # 图片保存路径
14. images_path = os.path.join(folder_path, 'input')
15. os.makedirs(images_path, exist_ok=True)
16.
17. ffmpeg_path = os.path.join(current_path, 'external', r'ffmpeg/bin/ffmpeg.exe')
18.
19. # 脚本运行
20. # 视频切分脚本
21. command = f'{ffmpeg_path} -i {video_path} -qscale:v 1 -qmin 1 -vf fps={fps} {images_path}\'\'
    %04d.jpg\'
22.
23.
24. subprocess.run(command, shell=True)
25. # COLMAP 估算相机位姿
26. command = f'python convert.py -s {folder_path}'
27. subprocess.run(command, shell=True)
28. # 模型训练脚本，模型会保存在 output 路径下
29. command = f'python train.py -s {folder_path}'
30. subprocess.run(command, shell=True)
```

代码 1 3D Gaussian 视频数据处理代码

三、引用

- [1] Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., ... Peng, S. (2024). Street Gaussians for Modeling Dynamic Urban Scenes.
- [2] Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4). Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., ... Peng, S. (2024). Street Gaussians for Modeling Dynamic Urban Scenes. arXiv preprint arXiv:2401.01339.
- [3] Zhou, H., Shao, J., Xu, L., Bai, D., Qiu, W., Liu, B., ... Liao, Y. (2024). HUGS: Holistic Urban 3D Scene Understanding via Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 21336-21345).