复现论文题目：3DGaussian Splatting for Real-Time Radiance Field Rendering

1. 研究背景

近年来，随着 NeRF（NeuralRadianceFields）[2]的提出，基于体渲染的 3D 场景重建和新视角合成技术取得了显著进展。NeRF 通过将场景表示为隐式的密集体素网络，利用射线采样和体积渲染的方式生成高质量的图像。然而，尽管 NeRF 在生成逼真的新视角图像方面表现优异，但它在实际应用中面临几个关键挑战：

训练时间过长：NeRF 通常需要数小时甚至数天的时间来完成训练，这使得其在需要快速部署的场景中应用受到限制。

实时渲染困难：由于 NeRF 使用的体渲染方法需要对大量的 3D 空间进行密集采样和计算，这导致其推理速度较慢，难以实现实时的新视角渲染。

显存占用高：NeRF 模型在训练和推理过程中需要存储大量的特征和采样点，这对硬件资源的需求较高。

针对这些问题，论文提出了一种新的方法——3DGaussianSplatting，它采用高斯分布的点云表示来替代 NeRF 的密集体渲染方法，并通过光栅化渲染实现高效的 3D 场景重建和实时渲染。

2. 方法框架

2.1 核心思想

3DGaussianSplatting 的核心思想是将 3D 场景表示为一组离散的 3D 高斯分布。与 NeRF 不同，3DGaussianSplatting 采用点云渲染而非体渲染，每个点云点用一个高斯分布来表示其位置、颜色以及不确定性（uncertainty）。通过这种表示方法，可以大大提高渲染效率，实现实时的新视角图像合成。

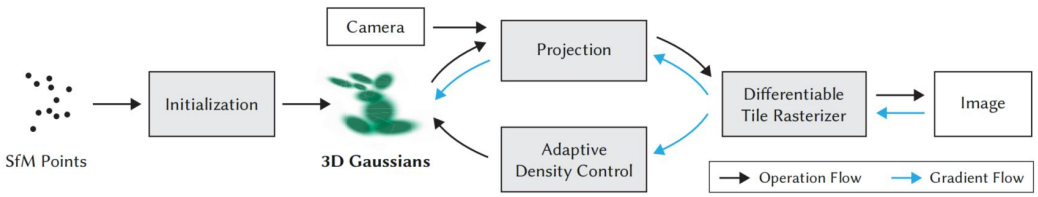


图 2.13D 高斯框架图

2.2 方法流程

Step1：3D 高斯点云的初始化

首先，从输入的多视角图像中通过传统的 SfM（Structure-from-Motion）方法生成一个初始稀疏点云。每个点云点被表示为一个 3D 高斯分布，并包含以下参数：

位置坐标：，表示点在 3D 空间中的位置。

颜色信息：RGB 值，表示该点的颜色。

协方差矩阵：表示点在空间中的尺度和不确定性，用于建模点的扩散程度。

Step2：高斯点云的优化

为了提高点云的重建精度，对初始化的 3D 高斯点云进行优化。优化的目标是调整每个高斯点的位置、颜色和不确定性参数，使渲染出的图像与原始输入图像之间的像素误差最小化。具体的优化步骤如下：

1）通过比较渲染图像与输入图像之间的差异来计算损失函数。

2）使用反向传播算法优化高斯点云的参数，使其更准确地表示原始场景。

Step3：光栅化渲染

与 NeRF 的体渲染不同，3DGaussianSplatting 使用光栅化渲染技术。该技术将高斯点云投影到图像平面，并根据高斯分布计算每个像素的颜色值。光栅化渲染的计算复杂度远低于体渲染，因此可以实现实时渲染。

Step4：多尺度融合

为了进一步提升渲染质量，该方法采用了多尺度融合策略。通过将不同尺度的高斯点云信息融合，可以更好地捕捉场景的全局结构和细节信息，从而生成更加逼真的新视角图像。

2.3 方法优势

相比于 NeRF，3D Gaussian Splatting 在多个方面表现出明显的优势：

训练速度快：由于使用点云的高斯表示和光栅化渲染，该方法的训练时间大大缩短，通常可以在 几分钟内完成训练。

实时渲染：得益于光栅化渲染技术，3D Gaussian Splatting 可以实现 实时的新视角合成，适用于虚拟现实、游戏和动画制作等场景。

高质量渲染：通过优化高斯点云的参数和多尺度融合策略，该方法在 图像质量 上能够与 NeRF 媲美，甚至在某些场景中表现更优。

2.4 技术对比

方法	训练时间	渲染速度	图像质量	显存占用
NeRF	数小时到数天	较慢	高	高
3D GS	几分钟	实时	高	低

表 2.1 技术对比表

通过表 2.1 对比可以看出，3D Gaussian Splatting 在训练时间和渲染速度上具有显著优势，特别是能够实现 实时渲染，这对于许多实际应用场景（如虚拟现实、动画制作等）具有重要价值。而且，该方法在显存占用方面更低，使其对硬件资源的需求更加友好。这些特性使得 3D Gaussian Splatting 成为传统 NeRF 方法的有效替代方案，尤其是在对实时性和硬件资源敏感的场景中。

3. 实验结果

3.1 实验设置

作者已经提供了项目源代码，链接如下：https://github.com/graphdeco-inria/gaussian-splatting.

1）实验框架：基于 PyTorch 框架，在单个 RTXA6000GPU 上进行实验，并根据 3D-GS 的配置微调优化参数。

```
parser = ArgumentParser(description="Training script parameters")
lp = ModelParams(parser)
op = OptimizationParams(parser)
pp = PipelineParams(parser)
parser.add_argument( *name_or_flags: '--ip', type=str, default="127.0.0.1")
parser.add_argument( *name_or_flags: '--port', type=int, default=6009)
parser.add_argument( *name_or_flags: '--debug_from', type=int, default=-1)
parser.add_argument( *name_or_flags: '--detect_anomaly', action='store_true', default=False)
parser.add_argument( *name_or_flags: "--test_iterations", nargs="+", type=int, default=[7_000, 30_000])
parser.add_argument( *name_or_flags: "--save_iterations", nargs="+", type=int, default=[7_000, 30_000])
parser.add_argument( *name_or_flags: "--quiet", action="store_true")
parser.add_argument( *name_or_flags: '--disable_viewer', action='store_true', default=False)
parser.add_argument( *name_or_flags: "--checkpoint_iterations", nargs="+", type=int, default=[7_000, 30_000])
parser.add_argument( *name_or_flags: "--start_checkpoint", type=str, default=None)
args = parser.parse_args(sys.argv[1:])
args.save_iterations.append(args.iterations)
```

图 3.13D-GS 配置[1]

2）数据集：**PKU-DyMVHumans[3]**

- 采集设备与场景

利用超过 60 个同步相机对不同场景进行数据采集，涵盖了 45 个多样化场景，确保能够捕捉到丰富的人类动态信息。这些场景包含各种日常活动以及复杂的动作姿态，以充分体现现实世界中人类行为的多样性。

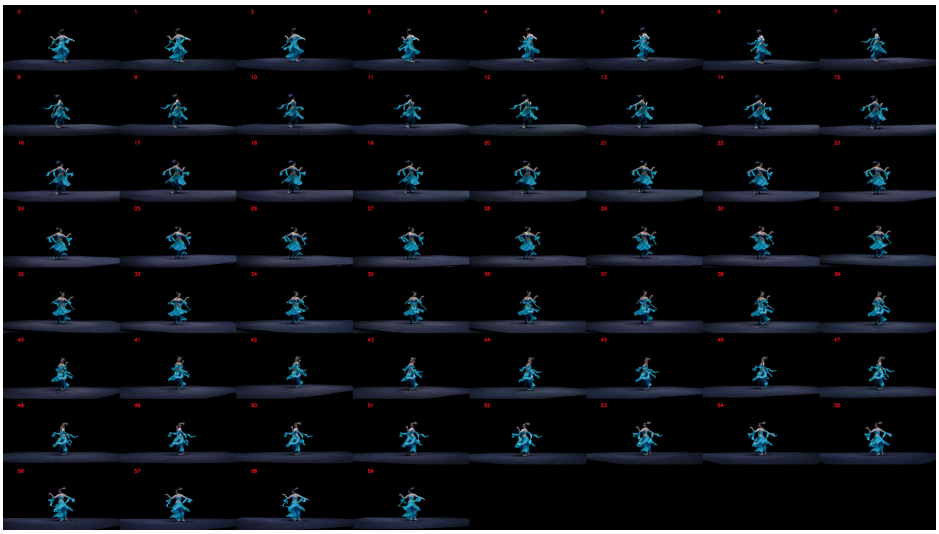


图 3.2PKU-DyMVHumans 数据集单帧多视角场景

- 数据内容与主体

数据集中包含 32 个人类主体，共计 820 万帧数据。每个主体具有高细节外观，且运动表现逼真，特别注重对宽松或超大服装以及复杂姿势的捕捉，以应对当前研究中的挑战。采集到的数据经过整理和预处理，以便于后续的实验使用。

- 数据集优势

有助于克服获取高质量人类数据集的挑战，为相关研究提供了可靠的数据来源，可用于训练和评估各种重建和渲染算法，提高算法在复杂场景下的性能。基于该数据集精心搭建了现成框架，方便与基于神经辐射场（NeRF）或 3D 高斯的最新场景表示方法结合，为基于 NeRF 和 3D 高斯的实现提供数据支持和基准测试，有助于推动相关算法在高保真动态数据处理方面的发展，如细粒度前景/背景分解、高质量人类重建和动态场景的

逼真新视图合成等应用。

3.2 实验结果评估指标

使用多种指标评估，包括峰值信噪比（PSNR）、感知质量度量 LPIPS、结构相似性指数（SSIM）及其扩展（如结构差异指数度量 DSSIM、多尺度结构相似性指数 MS-SSIM）、每秒帧数（FPS）、训练时间和存储。

各主要指标的定义和计算公式如下：

1）峰值信噪比 PSNR：是一种衡量图像质量的指标，用于比较原始图像和重建/渲染图像之间的差异。它基于均方误差（MSE）计算，MSE 是原始图像与处理后图像对应像素值之差的平方的平均值。PSNR 值越高，表示图像质量越好，重建或渲染的精度越高。

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right)$$

其中，MAX_I^2是图像中可能的最大像素值（对于 8 位灰度图像，$\mathrm{MAX}_I^2 = 255$）；MSE 是均方误差。

2）感知质量度量 LPIPS（LearnedPerceptualImagePatchSimilarity）：通过使用深度神经网络学习到的特征来度量图像之间的感知相似性。它试图模拟人类视觉系统对图像质量的感知，不仅仅关注像素级的差异，还考虑了图像的语义和结构信息。LPIPS 值越低，表示图像在感知上越相似，重建或渲染的质量越高。

$$\mathrm{d}(x, y) = \sum_l \frac{1}{H_l W_l} \sum_{h,\,w} \left| w_l \odot \phi_l(x)_{h,w} - w_l \odot \phi_l(y)_{h,w} \right|$$

其中，x 和 y 是要比较的两幅图像，ϕ_l是神经网络在第 l 层的特征映射，H_l和W_l是该层特征映射的高度和宽度，w_l是可学习的权重，$\odot$表示逐元素相乘。

3）结构性相似性指数（SSIM）：用于衡量两幅图像之间的结构相似性，它同时考虑了图像的亮度、对比度和结构信息。SSIM 值的范围在-1 到 1 之间，值越接近 1 表示两幅图像在结构上越相似，重建或渲染的图像在结构上越接近原始图像。

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

其中，x 和 y 是要比较的两幅图像，μ_x和μ_y分别是 x 和 y 的平均亮度，σ_x^2和σ_y^2是它们的方差，σ_{xy}是 x 和 y 的协方差，C_1和C_2是为了避免分母为零而添加的小常数。

3.3 原代码在 PKU-DyMVHumans 数据集上的结果

1）根据 COLMAP 得到的初始化点云

该模型输入是多个视角图像，共有 60 个相机视角，本实验采用前 50 个视角作为训练集，最后 10 个视角作为测试集。

初始化点云共有 4416 个点，可见经过 COLMAP 初始化，只是将人体部分的点云以及部分衣物的点云重建了出来，而忽视了场景地面。

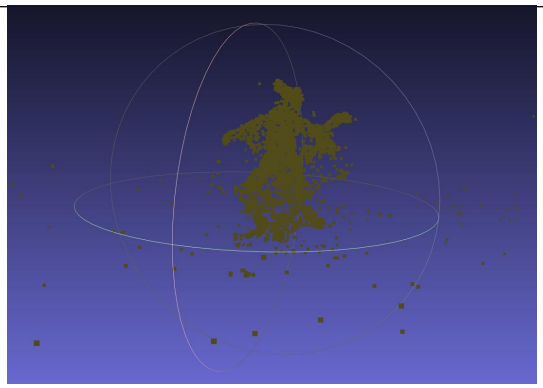


图 3.3 初始化点云

2）训练 30000 次迭代后的点云

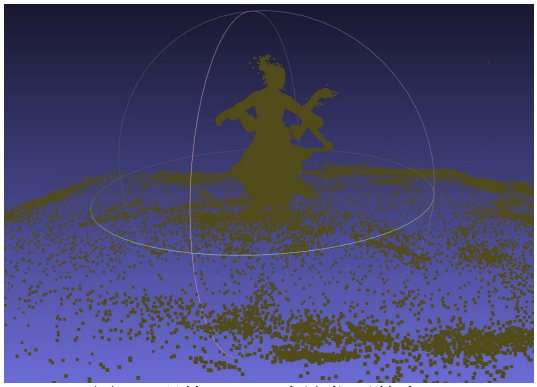


图 3.4 训练 30000 次迭代后的点云

经过网络迭代 30000 次后，模型将地面点云重建出来。

3）渲染后的训练集和测试集图像

4）

(a) GT **(b)渲染图像**

图 3.5 训练集图像

(b) GT **(b)渲染图像**

图 3.6 测试集图像

5）量化结果

指标	PSNR↑	LPIPS↓	SSIM↑
测试集	25.957	0.100	0.902

表 3.1 原代码测试结果

4. 讨论与改进思路

4.1 现有成果总结

高效的点云表示：3D GS 通过高效表示大规模的 3D 点云数据（例如，衣物和人体的表面）来生成复杂场景。它使用了 Gaussian Splats，即通过高斯函数来近似点云中的每个点，从而更好地捕捉形状和纹理信息。

多视角融合：该方法能够处理来自多个视角的数据，并能有效融合这些视角中的信息，以生成一致的三维重建模型。这对于复杂衣物人体重建尤其重要，因为不同视角提供了不同的信息，尤其是在衣物细节和人体姿态的重建中。

实时渲染：得益于高效的点云渲染方法，3D GS 支持实时渲染和交互，尤其在虚拟现实（VR）和增强现实（AR）应用中表现出色。

纹理合成：通过 Gaussian Splats 的技术，可以合成细节丰富的纹理，这对于衣物和人体表面细节的还原非常重要。

4.2 局限性分析

深度信息的缺乏：目前，3D GS 技术主要依赖于从多个视角获取的 RGB 图像，而缺乏深度图作为补充。这使得在处理复杂几何形状（如人体和衣物的褶皱）时，可能会出现误差和伪影。特别是在衣物的重建中，复杂褶皱和遮挡关系可能导致信息缺失或模糊。

视角一致性问题：尽管 3D GS 能够通过多视角进行重建，但当视角间的差异较大（例如不同的相机配置或不同的姿势时），可能会导致不一致的重建结果。由于缺乏深度图来提供更多的空间信息，这些不一致性可能会导致重建效果的下降，尤其是在细节层面。

计算复杂度和资源需求：3D GS 涉及大量的点云数据和复杂的高斯函数运算，因此在处理高分辨率数据时，计算资源的需求较高。这对于实时或大规模数据的处理可能是一个瓶颈，特别是在计算资源有限的情况下。

缺乏全局优化机制：现有的 3D GS 方法可能没有充分考虑全局的几何和纹理一致性，尤其是在处理不同角度和视角的融合时，可能会缺乏一种统一的优化机制来解决视角间的几何误差和纹理接缝问题。

4.3 改进思路

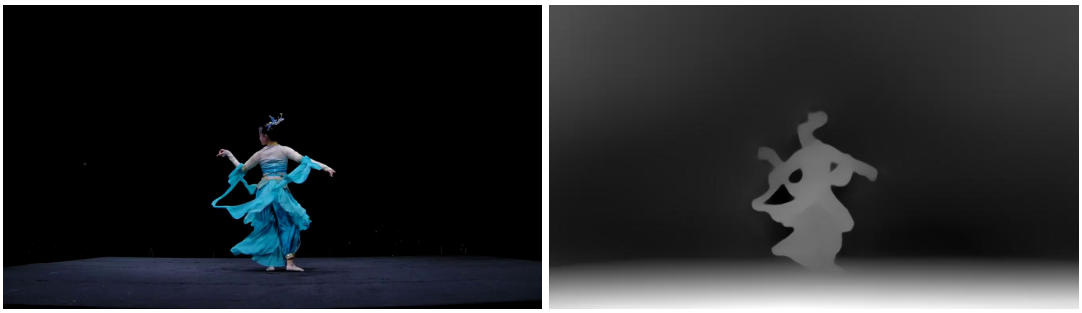
为克服上述局限性，计划在现有方法基础上进行改进。鉴于数据集缺乏深度图信息，拟引入现有的深度计算模型来获取视角图像的深度信息。具体而言，将选择合适的深度估计模型，例如基于深度学习的单目深度估计网络，将其应用于输入的视角图像，从而生成对应的深度图。深度图的引入将为后续处理提供更多几何信息，有助于更准确地建模场景结构与物体运动。在处理人体服装重建时，深度图可辅助更好地理解服装的三维形态及其在运动过程中的变化，从而更精确地模拟服装的形变和运动轨迹。同时，深度信息还可用于优化相机姿态估计，提高系统对复杂场景的理解与处理能力，进一步提升

整体重建与渲染质量，推动该方法在人体服装重建及其他相关领域的应用发展。

4.4 深度估计模型

MiDaS（Monocular Depth Estimation via a Multi-Scale Network）[4]是一种先进的单目深度估计模型，专门用于从单张图像中估算场景的深度信息。其核心创新是使用多尺度网络架构，通过不同尺度的特征提取，有效地捕捉图像中的深度信息，提升了深度估计的精度。MiDaS 不仅能处理常见的自然场景，还能适应室内外环境，能够在各种条件下生成高质量的深度图。该模型的训练通常是在多个大规模数据集（如 NYU Depth V2 和 KITTI 等）上进行的，因此它具有较强的泛化能力，能够应对不同的场景和环境。这使得 MiDaS 能够广泛应用于增强现实（AR）、虚拟现实（VR）、自动驾驶、机器人视觉等多个领域。例如，在自动驾驶中，MiDaS 为实时深度估计提供支持，帮助车辆识别道路和障碍物。在增强现实中，它可以为虚拟物体与现实场景的融合提供准确的深度信息。

然而，MiDaS 也存在一些局限性。首先，尽管它可以生成精确的深度图，但依赖图像中的纹理信息，使得在纹理较少的区域（如平坦的墙面或地面）深度估计的精度可能较低。此外，由于 MiDaS 模型较为复杂，它的推理速度可能较慢，这在一些需要实时反馈的应用中可能会成为瓶颈。另一方面，MiDaS 主要设计用于静态场景，对于动态场景中的快速运动物体或变化，它的表现可能不如专门为动态场景优化的深度估计模型。尽管如此，MiDaS 依然在深度估计领域展现出强大的能力，特别是在处理静态和纹理丰富的场景时，能够提供高精度的深度图，对于复杂的三维重建任务，如多视角复杂衣物人体重建，MiDaS 的深度信息能够有效帮助提高重建精度，减少伪影的产生。因此，它仍然是一个非常有价值的工具，特别是在需要高质量深度估计的应用中。



(a) 原图 **(b) 深度预测图像**

图 4.1 单帧单视角深度预测

需要添加深度图之处可由图 4.2 中的这段代码给出：

这段代码主要的目的是处理 COLMAP 中与深度图相关的参数文件，depth_params.json，并进行一些必要的检查和处理。

首先，代码会检查指定路径下是否存在 depth_params.json 文件。如果该文件存在，它会被加载并解析为一个字典对象。然后，代码从文件中提取出所有深度图的缩放因子（scale），并计算出非零缩放因子的中位数（med_scale）。这个中位数将被添加到每个深度图的参数中，作为一个标准化的标尺，可能用于后续处理深度图的统一尺度。如果 depth_params.json 文件不存在，或者在尝试加载文件时发生错误，代码会给出错误提示并终止程序。

如果存在深度图参数文件并且深度图的相关数据有效，代码还会对深度图的缩放因子进行处理，确保不同图像间的深度值有统一的尺度。

从这段代码可以看出，深度图（及其缩放因子）是计算 3D 重建和深度估计的关键。

COLMAP 依赖深度图来恢复场景的几何结构，并且深度图的缩放因子需要被标准化和统一，这对于后续的重建过程至关重要。没有这些参数，尤其是深度图的中位数尺度，可能会导致深度估计和图像重建中的不一致性或错误，从而影响重建的精度和效果。

因此，添加深度图可以为重建过程提供更多的空间信息，帮助在多视角重建时提高精度，并减少因缺失深度信息而导致的伪影问题。

```
depth_params_file = os.path.join(path, "sparse/0", "depth_params.json")
## if depth_params_file isnt there AND depths file is here -> throw error
depths_params = None
if depths != "":
    try:
        with open(depth_params_file, "r") as f:
            depths_params = json.load(f)
        all_scales = np.array([depths_params[key]["scale"] for key in depths_params])
        if (all_scales > 0).sum():
            med_scale = np.median(all_scales[all_scales > 0])
        else:
            med_scale = 0
        for key in depths_params:
            depths_params[key]["med_scale"] = med_scale

    except FileNotFoundError:
        print(f"Error: depth_params.json file not found at path '{depth_params_file}'.")
        sys.exit(1)
    except Exception as e:
        print(f"An unexpected error occurred when trying to open depth_params.json file: {e}")
        sys.exit(1)
```

图 4.2 加入深度图

如要符合生成 depth_params.json 文件的需要，本文采用以下图 4.3 中的代码生成：MiDaS 生成的深度图是以 NumPy 数组的形式输出的。在保存时，这些深度图将被转换为列表格式，并与相机的内参一起存储在 depth_params.json 文件中。这样做的目的是为了确保深度图可以与相机的标定参数一起使用，便于后续的深度重建和三维处理。

```
def save_depth_params(depth_map, camera_intrinsics, output_dir):
    """Save depth map and camera parameters to a JSON file compatible with COLMAP format.
    Args:
        depth_map (numpy.ndarray): The depth map generated by MiDaS.
        camera_intrinsics (dict): A dictionary containing the camera's intrinsic parameters.
        output_dir (str or Path): The directory where the depth_params.json file will be saved."""
    # Ensure the output directory exists
    output_dir = Path(output_dir)
    output_dir.mkdir(parents=True, exist_ok=True)
    # Define COLMAP-style depth parameters
    depth_params = {
        "depth_map": depth_map.tolist(),  # Convert numpy array to list for JSON compatibility
        "camera": {
            "fx": camera_intrinsics["fx"],  # Focal length in x direction
            "fy": camera_intrinsics["fy"],  # Focal length in y direction
            "cx": camera_intrinsics["cx"],  # Principal point in x direction
            "cy": camera_intrinsics["cy"],  # Principal point in y direction
            "width": camera_intrinsics["width"],  # Image width
            "height": camera_intrinsics["height"],  # Image height
        },
        "depth_scale": 1.0}
    # Define the output file path
    depth_params_path = output_dir / "depth_params.json"
    # Save to JSON
    with open(depth_params_path, 'w') as json_file:
        json.dump(depth_params, json_file, indent=4)
    print(f"Depth parameters saved to: {depth_params_path}")
```

图 4.3 生成对应的 json 文件

相机内参包括焦距（fx 和 fy）、主点坐标（cx 和 cy）以及图像的宽高（width 和 height）。这些参数通常是通过相机标定过程获得的，确保了深度图与相机成像系统的一致性。在

保存深度图时，您需要提供这些参数，以便为深度估计过程提供必要的几何信息。

保存路径由 output_dir 指定，这个目录可以根据您的需求进行调整。最终，depth_params.json 文件将被保存在该目录下，确保深度图和相机内参的存储位置一致，便于后续访问和处理。

需要特别注意的是，生成的 depth_map 必须与相机的图像尺寸相匹配。例如，MiDaS 输出的深度图应与原始图像的分辨率一致，以确保深度信息的准确性。另外，depth_scale 是一个可选参数，它用于设置深度图的缩放比例，您可以根据实际情况进行调整，以适应不同的深度尺度要求。

4.5 改进后的实验结果

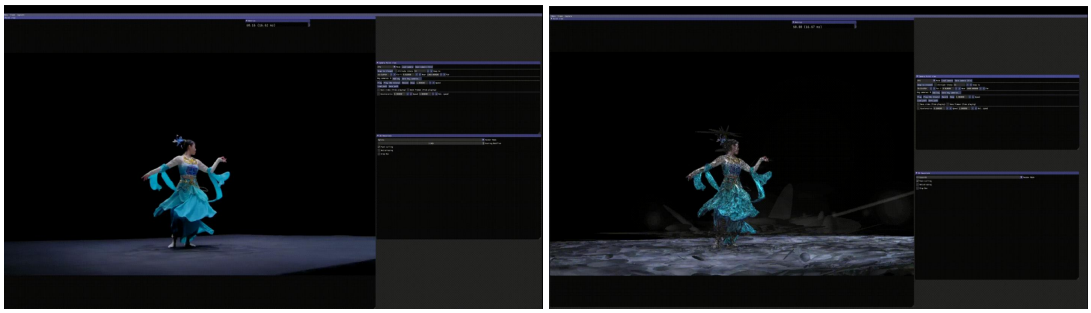
1）可视化结果

同样的，针对第一帧图像，渲染结果如下图 4.4 所示：

(a) GT **(b) 渲染图像**

图 4.4 单帧单视角深度预测

并且，采用 SIBR 可视化浏览器后，可以看到多个视角的 3D Splatting 和椭球显示，如下图 4.5，如若需要查看动图，可以点击链接：https://github.com/BruceYounggit/3DGS/tree/main/demos。



(a) Splatting 显示 **(b) 高斯椭球显示**

图 4.5 SIBR Viewer 显示

2）量化结果

指标	PSNR↑	LPIPS↓	SSIM↑
测试集	27.123(↑1.166)	0.097(↓0.003)	0.932(↑0.03)

表 4.1 改进后测试结果

由表 4.1 可知，改进后的结果显示，引入深度图显著提升了模型的重建效果。PSNR 的提升表明图像质量更高，重建图像与原图差异减少。LPIPS 的下降意味着重建图像在人眼感知上更为相似，视觉差异减小。SSIM 的提高则表示图像的结构和细节更加一致，整体重建效果更加稳定和精确。总体来看，深度图的加入有效提高了图像的清晰度、感知质量和结构一致性。

5. 总结

在进行复杂衣物人体的三维重建时，本文选择了基于 3D 高斯（3D Gaussian Splatting）的方法，并针对单帧的多视角场景进行了重建。这样做的原因有两个主要方面。

首先，3D 高斯建模方法对于处理复杂形状和高纹理场景非常有效，特别是在需要高精度的三维重建时。通过将物体表面表示为多个高斯点（或体素），我们可以在保留细节的同时有效地进行表面拟合，从而减少因纹理复杂或细节不足导致的重建误差。对于衣物和人体等具有高度动态形变和细节的场景，3D 高斯模型能够提供更为稳定和精确的重建效果，尤其是在动态变化或复杂遮挡的情况下。

其次，我们选择了单帧多视角的方案，这是因为在大多数实际应用场景中，获取大量连续的视角数据可能是困难的，尤其是在高复杂度的物体或场景中。单帧多视角通过不同角度的视图合成一个综合的三维信息模型，能够在有限的输入条件下获得较为完整的重建结果。尽管该方法在某些细节上可能不如多帧视频的方式精细，但它在处理复杂衣物人体这类动态场景时，能够提供较好的平衡。

然而，深度图的引入为改进方案带来了显著的提升。深度图提供了每个像素点到相机的距离信息，它直接帮助了我们在三维空间中定位每个像素的深度位置。通过引入深度图，我们能够更加精确地约束和优化三维重建过程，尤其是在解决物体遮挡、细节捕捉和空间错位等问题时。

具体来说，添加深度图后的效果有几个明显的改进：

提高重建精度：深度图通过为每个视角提供深度信息，帮助我们更准确地复原场景的空间结构，尤其是在复杂的衣物和人体形状的细节方面，深度信息补充了我们模型中缺失的几何数据。

减少伪影：在没有深度图的情况下，某些视角的重建结果可能会出现伪影或形状不准确的问题。深度图为每个像素提供了一个具体的空间位置，有效避免了由于多视角融合产生的不一致性和伪影现象。

增强多视角合成效果：通过深度图的辅助，多个视角的信息可以更好地对齐和融合，减少了重建中的模糊和畸变，进一步提高了多视角合成的质量。

总结而言，虽然 3D 高斯方法本身在单帧多视角的条件下能够提供合理的重建效果，但深度图的加入显著提升了重建的精度和稳定性，尤其是在处理复杂衣物人体重建这种具有较强纹理和几何挑战的任务时，它有效减少了伪影并优化了三维结构的精准度。

参考文献：

[1] KerblB,KopanasG,LeimkühlerT,etal.3dgaussiansplattingforreal-timeradiancefieldrendering [J].ACMTrans.Graph.,2023,42(4):139:1-139:14.

[2] Mildenhall B, Srinivasan P P, Tancik M, et al. Nerf: Representing scenes as neural radiance fields for view synthesis[J]. Communications of the ACM, 2021, 65(1): 99-106.

[3] Zheng X, Liao L, Li X, et al. PKU-DyMVHumans: A Multi-View Video Benchmark for High-Fidelity Dynamic Human Modeling[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 22530-22540.

[4] Ranftl R, Lasinger K, Hafner D, et al. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer[J]. IEEE transactions on pattern analysis and machine intelligence, 2020, 44(3): 1623-1637.