Supplementary Material For SCRREAM

HyunJun Jung
Technical University of Munich
`hyunjun.jung@tum.de`

Weihang Li
Technical University of Munich

Shun-Cheng Wu
Technical University of Munich

William Bittner
Technical University of Munich

Nikolas Brasch
Technical University of Munich

Jifei Song
Huawei Noah's Ark Lab

Eduardo Pérez-Pellitero
Huawei Noah's Ark Lab

Zhensong Zhang
Huawei Noah's Ark Lab

Arthur Moreau
Huawei Noah's Ark Lab

Nassir Navab
Technical University of Munich

Benjamin Busam
Technical University of Munich
`b.busam@tum.de`

1 Hardware Setup

We use a specially designed camera rig (Fig. 1, (a)) to capture the synchronized multi-modal image sequences in a free hand manner. RGB and polarization images are acquired with a Phoenix 5.0 MP Polarization camera (PHX050S1-QC, LUCID Vision Labs, Canada) equipped with a Sony Polarsens sensor (IMX264MYR CMOS, Sony, Japan). For the depth images, we use two depth modality, active stereo depth and indirect time-of-flight depth. Intel RealSense D435 (Intel, USA) is used for the active stereo depth and Lucid Helios (HLS003S-001, LUCID Vision Labs, Canada) is used for indirect time-of-flight depth. For synchronization, a Raspberry Pi generates a trigger signal that is connected to all cameras to trigger the image acquisition.

Two different types of scanners are used for the scanning. For the small household objects, such as cups or water bottles, we use the table-top scanner (EinScan-SP, SHINING 3D Tech. Co., Ltd., Hangzhou, China, Fig. 1, (b)) to capture the meshes with fine details. For the larger objects that do not fit to the table-top scanner, such as furniture or medium size objects or room, we use hand held scanner (Artec-Leo, Artec 3D, Luxembourg, Fig. 1, (c)) that runs SLAM internally for tracking the scanning. To scan large texture-less areas, such as walls or big tables, we use small stripes of masking tape on the surface to provide additional texture (Fig. 2, (a)) during the scanning. Later, the textures from the masking tapes are removed via the texture healing function in Artec Studio 17 ((Fig. 2, (b)).

2 Dataset Examples

Indoor Reconstruction and SLAM Dataset. We provide 11 scenes that comprise 7114 frames, 94 object & furniture meshes and 7 indoor room meshes for the indoor reconstruction and SLAM dataset. Note that all furniture and objects in the scenes are scanned prior to ensure the meshes are high resolution and water-tight, in order to serve as digital twin assets so that the rendered depth does not miss any parts of the scene. The visualization of each scene mesh is shown in Fig. 3.

38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks.

(a) Camera Rig   (b) Shining 3D EinScan   (c) Artec Leo Hand-Held Scanner

Figure 1: **Hardware Setup.**

(a) Wall with masking tape patches as a texture   (b) Scanned mesh after texture healing via ArtecStudio 17

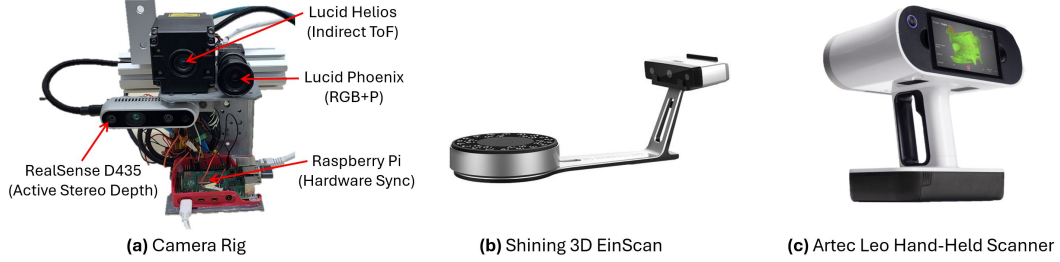
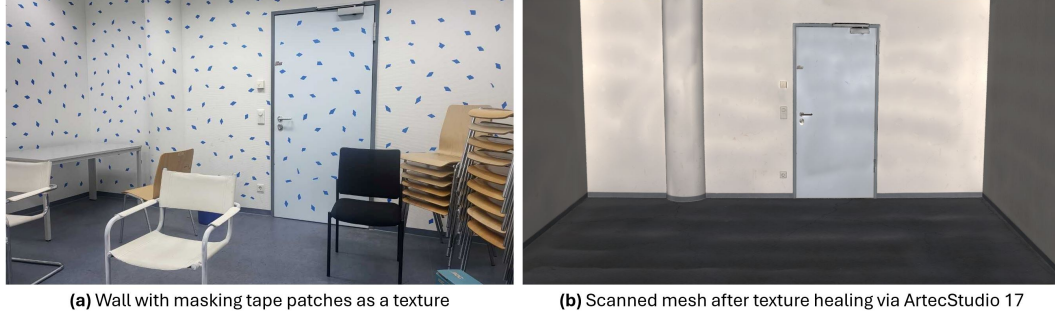



Figure 2: **Example of Wall Scanning.** To help tracking the hand-held scanner on the plain white wall, we attach patches of masking tape on the wall to provide additional texture (a). Later, the texture of the tape can be removed in a post-processing step (b).

Object Removal & Scene Editing Dataset. We capture multiple additional video sequences for each of the 8 selected scenes from the indoor reconstruction and SLAM dataset each with a reduced number of objects or furniture to obtain multiple examples for object removal and scene editing tasks. This set is comprised of additional 9323 frames. A visualization of each reduced scene is shown in Fig. 4- 11. Note that for each reduced scene, we capture the real image sequence with the camera rig.

Human Reconstruction Dataset. For the semi-dynamic human sequences with the mannequin, we do scanning, registration, rendering and mapping for all individual frames. We showcase 2 scenes captured with this setup. We show the scene with a single human mesh from the first frame (Fig 12, left) to show the scene setup as well as all human meshes from the entire video sequence (Fig 12, right) to show the motion range of the mannequin. Note that around 4 real images are captured per time frame for the mapping step.

6D Pose Estimation Dataset. We showcase two pose dataset scenes in Fig 13. By nature of our dataset annotation pipeline, we can obtain object poses w.r.t. the world coordinate system $T_{obj->world}$ and camera poses w.r.t. the world coordinate system $T_{cam->world}$. By concatenating both pose in the right order (e.g. $T_{obj->world}^{-1} \cdot T_{obj->world}$), the object pose w.r.t. the camera center $T_{obj->cam}$, the so called 6D object pose, can be obtained. Having registered the object poses using our framework, the free-hand camera rig can be used to quickly and easily capture a pose dataset with extensive camera coverage.

2

Scene01   Scene02   Scene03

Scene04   Scene05   Scene06

Scene07   Scene08   Scene09

Scene10   Scene11

Figure 3: **Example of Indoor Reconstruction and SLAM Scenes.**

scene01_full_00   scene01_reduced_00

Figure 4: **Example of Reduced Scenes For Scene01.**

3

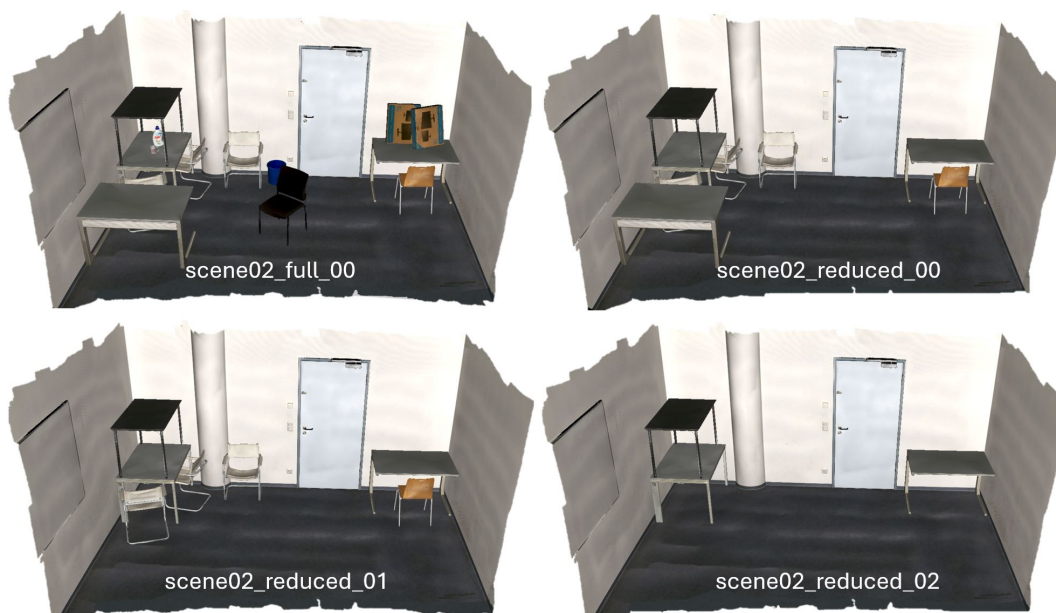



Figure 5: **Example of Reduced Scenes For Scene02.**

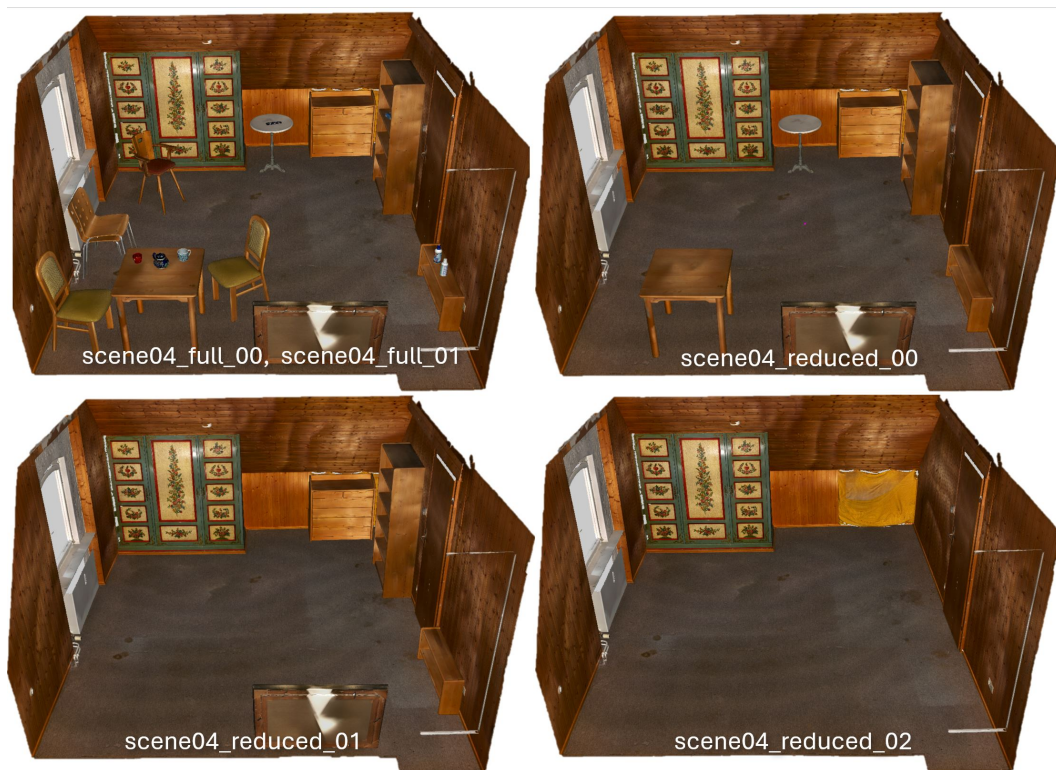



Figure 6: **Example of Reduced Scenes For Scene04.**

4

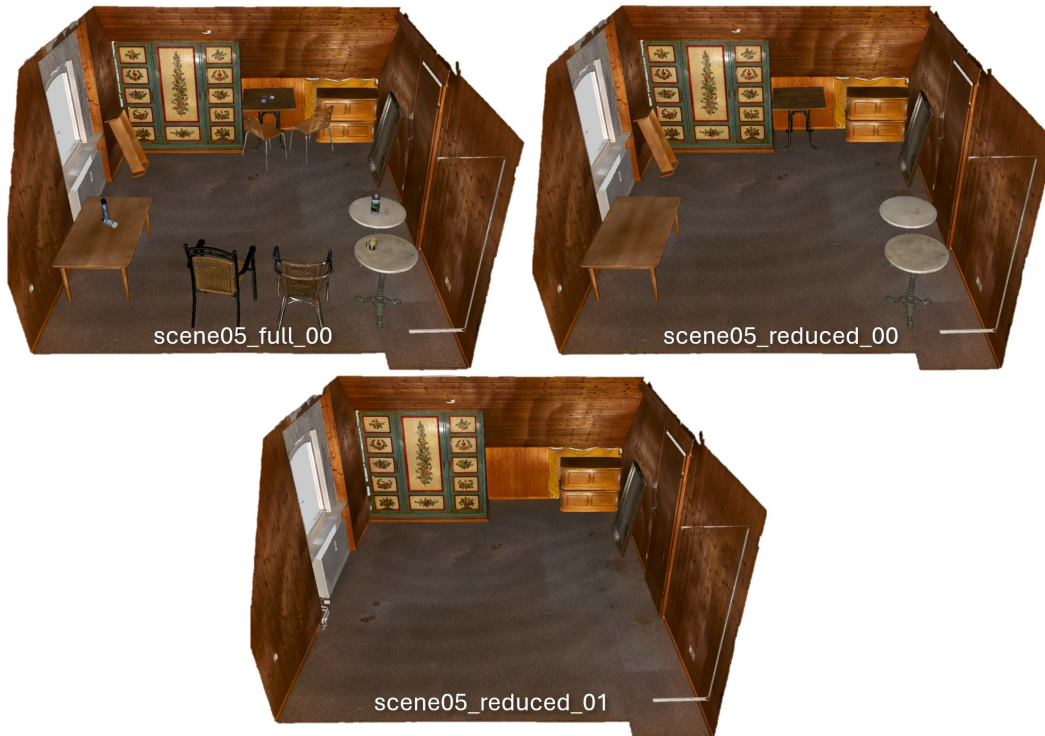



Figure 7: **Example of Reduced Scenes For Scene05.**

Figure 8: **Example of Reduced Scenes For Scene06.**

5

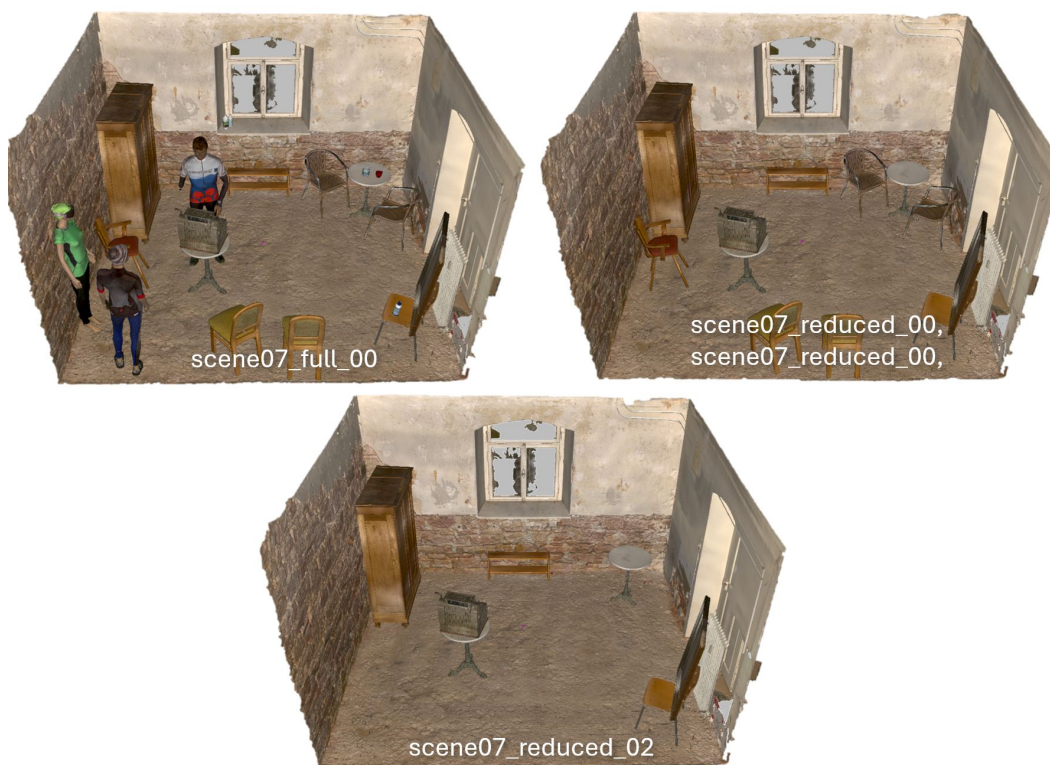



Figure 9: **Example of Reduced Scenes For Scene07.**

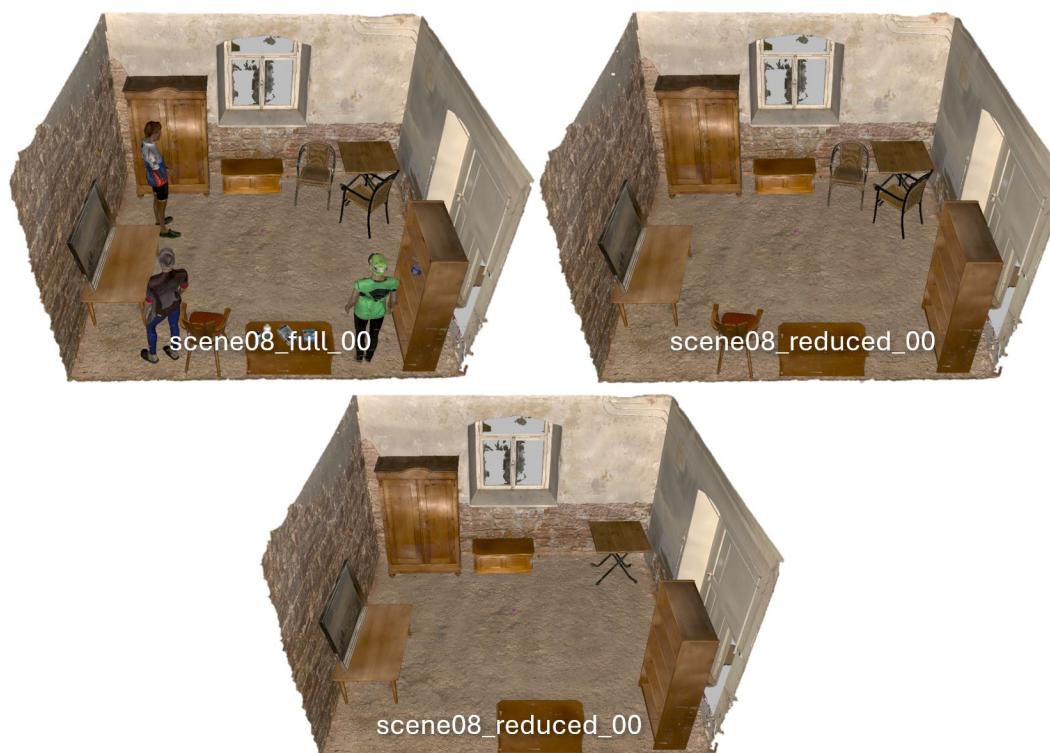



Figure 10: **Example of Reduced Scenes For Scene08.**

6

scene09_full_00

scene09_reduced_00

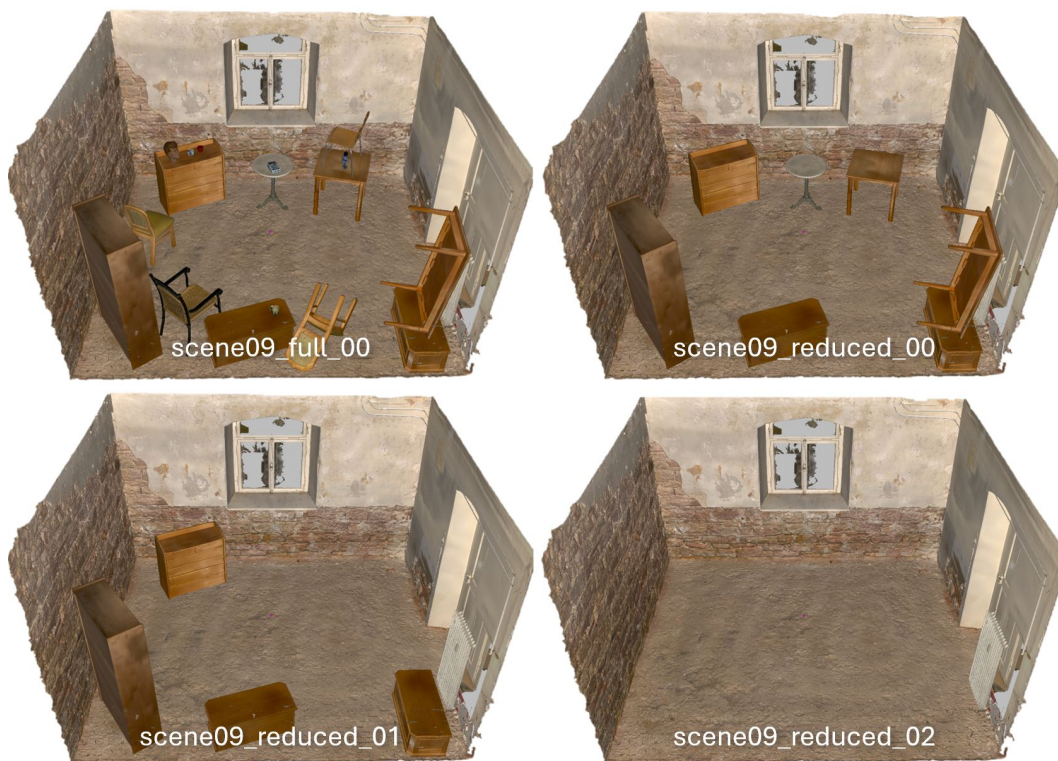
scene09_reduced_01

scene09_reduced_02

Figure 11: **Example of Reduced Scenes For Scene09.**

7

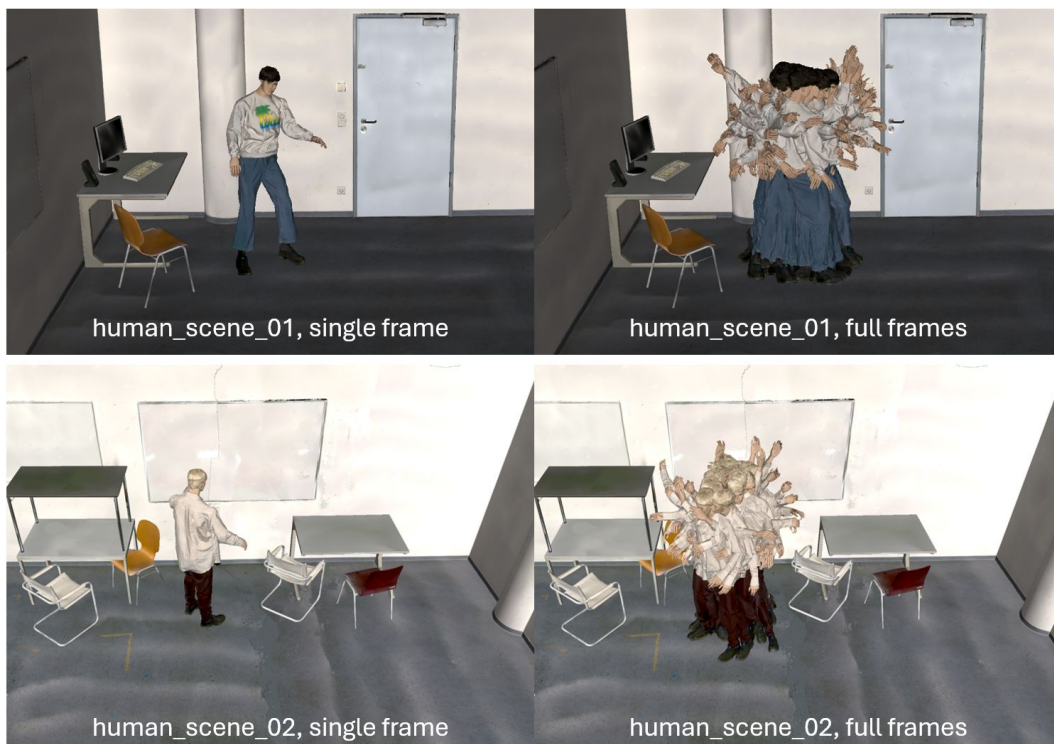



Figure 12: **Example of Human Dataset Scenes.**

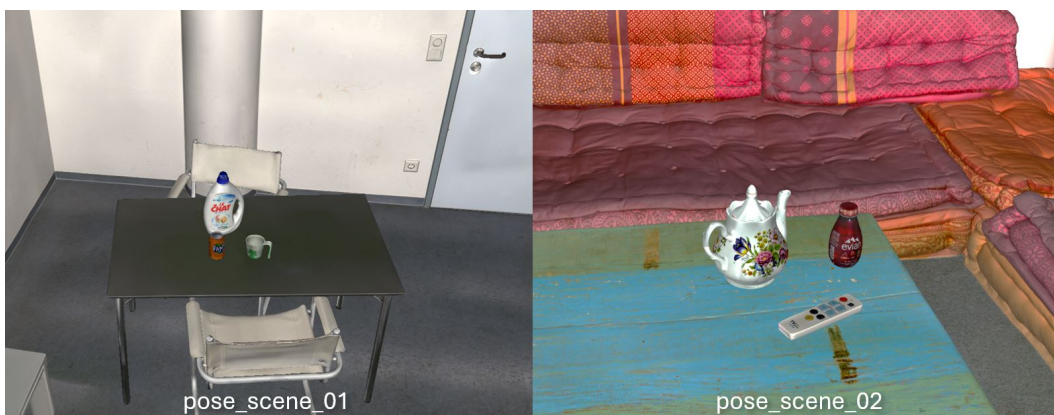



Figure 13: **Example of Pose Dataset Scenes.**

8

3 Dataset Comparison

In this section, we show more in-depth comparisons between the existing indoor datasets and our dataset using meshes of the scanned scenes. We choose the ScanNet++ [1] and Replica dataset [2] for comparison.

The ScanNet++ dataset is an improved version of ScanNet [3]. Instead of using a commercial depth sensor like its predecessor, ScanNet++ uses a LiDAR sensor that rotates and captures a more accurate 360° pointcloud of the scene. To be less dependent on the sensor's line-of-sight ScanNet++ captures the scene from multiple locations in the scene in order to improve the quality and completeness of the mesh. However, no post-processing is applied, such as hole filling, so that the final mesh still has many missing parts, limiting the use of the scene meshes as ground truth for rendering complete views. This is because for the areas with missing geometry the background is visible in the rendering instead of the foreground such that a metric using the incomplete mesh as ground truth will penalize a method that models the foreground object correctly. We show in Fig. 14 that our dataset provides the full details on the office furniture such as the chairs, the trash bin, the monitor etc while ScanNet++ suffers from incomplete and missing parts in the mesh.

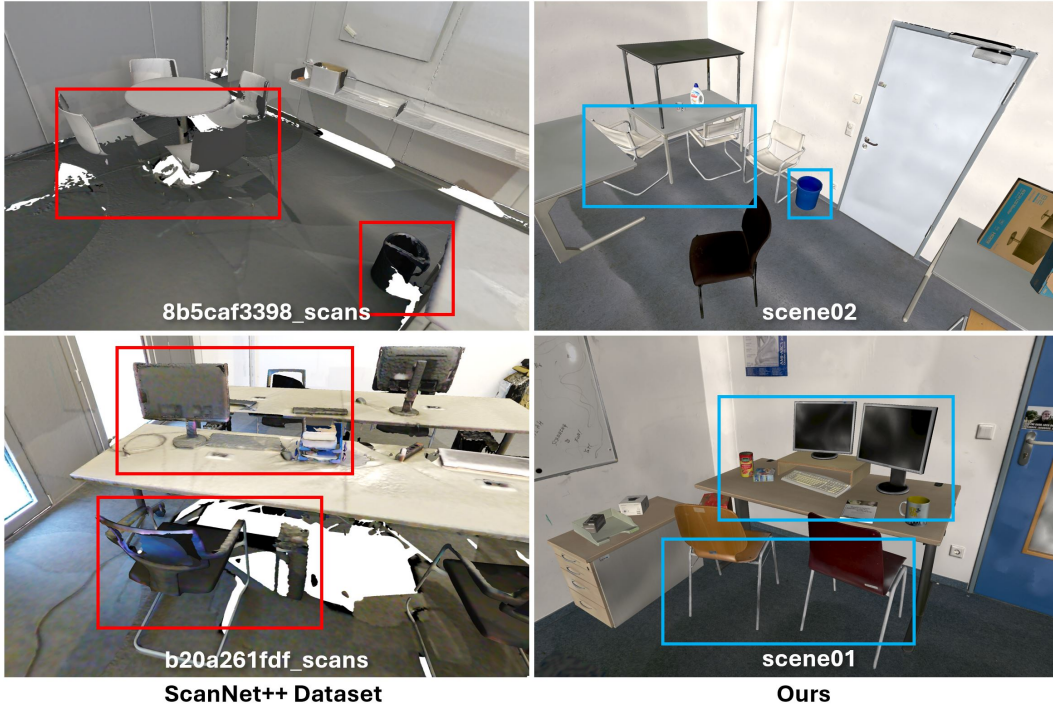



Figure 14: **ScanNet++ Dataset VS Ours.**

9

Unlike ScanNet++, the Replica dataset scans the scene with a specially designed camera rig that contains multiple sensors to obtain the mesh in more detail. Also, after the scanning, a manual hole-filling process is used to complete the mesh if the missing area is part of a flat surface. For this reason, the meshes of the Replica dataset are much more complete compared to ScanNet++. However, we find that meshes in the Replica dataset in general are overly smoothed and lack details. Also there are parts missing on non-flat surfaces, such as the legs of a chair or the tab of the sink. In Fig. 15 we provide a comparison between the Replica dataset and our dataset. We show a kitchen scene from each dataset to demonstrate that compared with Replica our dataset contains all small details of the scene, such as the small handles of the kitchen cabinet (zoom in for more details). We also show that the legs of the chairs and table are missing in a living room scene from the Replica dataset, while our dataset contains these parts in a high quality. We also attach in detailed comparison of the datasets in our supplementary video file.

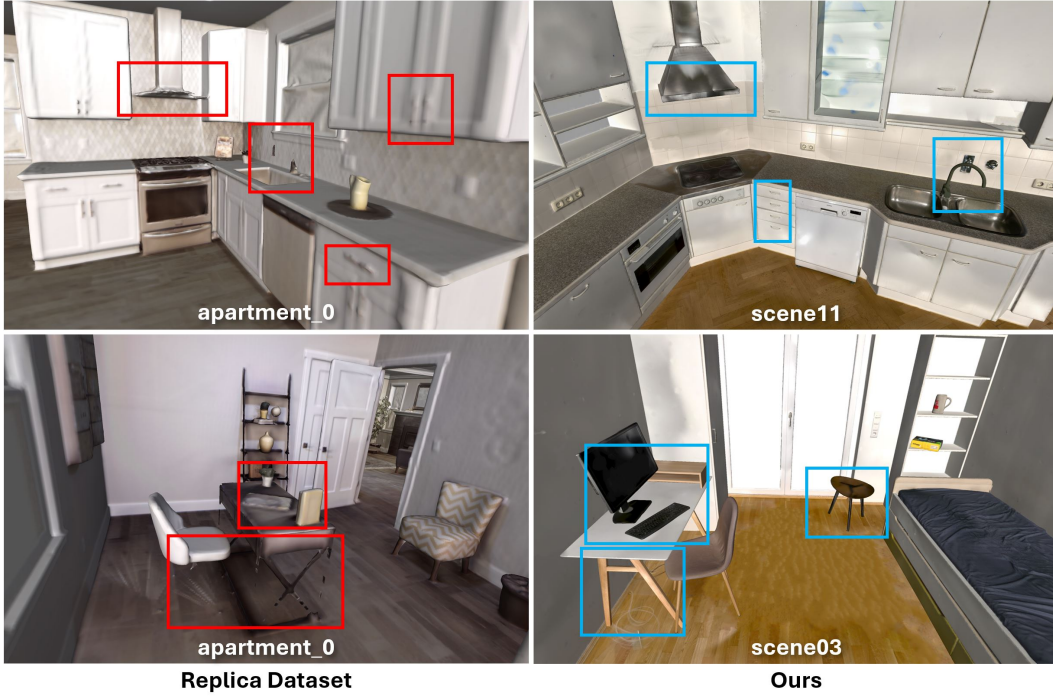



Figure 15: **Replica Dataset VS Ours.**

10

4 Dataset Documentation

Our dataset includes scene folders with 3 different conventions. Folders with the `"sceneXX"` convention contain data for the `Indoor Reconstruction and SLAM Dataset` and `Object Removal and Scene Editing Dataset`, folders with labelled `"human_sceneXX"` contain the `Human Reconstruction Dataset` and folders named `"pose_sceneXX"` contain the `6D Pose Estimation Dataset`. In this section, we explain the file structure of the dataset and describe the naming conventions for individual files in the folders. To access the dataset's download link as well as python code for data loading and visualization, please visit `https://github.com/Junggy/SCRREAM`.

4.1 Indoor Reconstruction and SLAM Dataset and Object Removal and Scene Editing Dataset.

Both the `Indoor Reconstruction and SLAM Dataset` and the `Object Removal and Scene Editing Dataset` follow the same format. The `"sceneXX"` folders contain 3 sub-folders: `"meshes"`, `"sceneXX_full_XX"` and `"sceneXX_reduced_XX"`.

The `"meshes"` folder contains independent meshes of the full scene that are aligned w.r.t. world coordinate system as `"*.obj"` files. Therefore loading all meshes together shows the full scene in the world coordinate system. Each mesh file follows a naming convention of `"{class_name}-{instance_name}.obj"`. For example, a monitor object with of the old_tall instance is named `"monitor-old_tall.obj"`. This naming convention allows easy identification of both class and instance of the objects in the scene.

The `"sceneXX_full_XX"` and `"sceneXX_reduced_XX"` folders contain the data regarding the image sequences, such as `"camera_pose"`, `"depth_d435"`, `"depth_gt"`, `"depth_tof"`, `"instance"`, `"pol"`, `"rgb"`, `"sparse"`, `"intrinsics.txt"`, `"meta.txt"` and `"video.avi"`. The detailed folder layout is shown in Fig. 16. Both full scene and reduced scene have the same file structure. Metadata from `"meta.txt"` can be used to identify the removed objects in the reduced scenes.

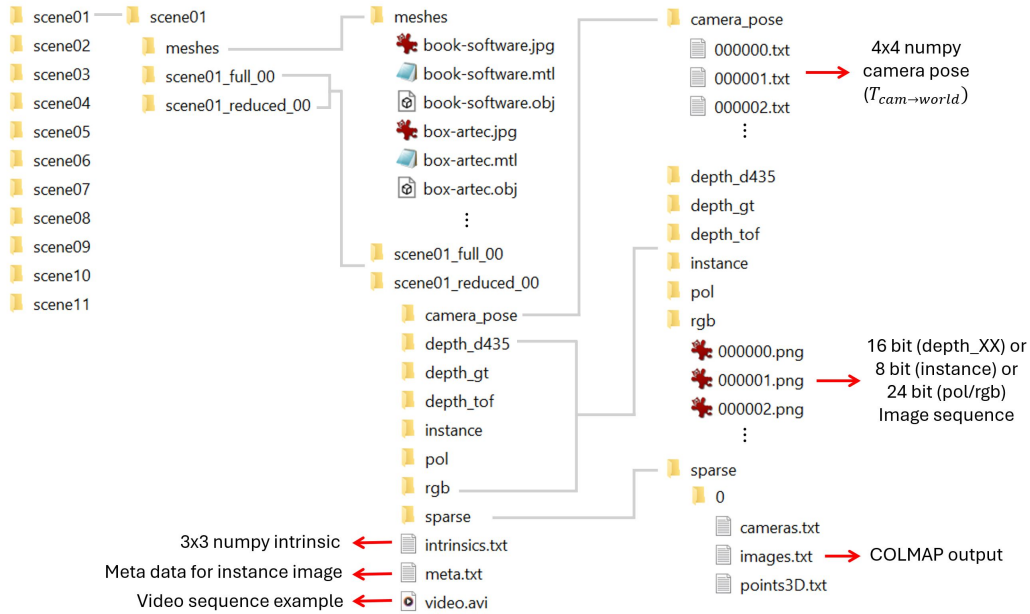



Figure 16: **File Structure of the Indoor Reconstruction and SLAM Dataset and Object Removal and Scene Editing Dataset.**

11

The `"camera_pose"` folder contains the camera pose for each image frame in the 4x4 matrix format saved as numpy matrix in a text file. The camera pose's definition is $T_{cam->world}$, meaning that each frame's translation element refers to the camera center in the scene. The matrix can be loaded with the `"np.loadtxt"` function.

The `"depth_d435"`, `"depth_gt"`, `"depth_tof"` folders contain the depth images for each image frame. Each depth image is saved as a 16bit image in mm units meaning that pixel value 1000 represents 1 meter. The depth images can be loaded in meter units by using opencv-python's `"cv2.imread"` function, `"cv2.imread("XXXXXX.png",-1).astype(np.uint16)/1000"`. Images from both the `"depth_d435"` and `"depth_tof"` folders are obtained from the real depth sensors and warped or aligned to the RGB image in a forward manner (e.g. forward-warping) using their own depth values, the extrinsic calibration matrix and RGB sensor's intrinsic matrix. Note that due to sensor noise, the warped depth are often not well aligned to the corresponding RGB image. Images in the `"depth_gt"` folder are the ground truth depth maps that are rendered from the object meshes and camera poses and serve as absolute ground truth for the depth evaluation benchmark.

The `"instance"` folder contains the corresponding instance segmentation images for the given RGB frames. The segmentation is obtained by rendering the individual objects with a unique value using the camera pose and intrinsic matrix. The mapping between the pixel values and object instances is provided in the `"meta.txt"` file. We keep the same mapping between the full and reduced scenes for convenience.

The `"meta.txt"` file contains the metadata that is mapping between the instance pixel values and the object meshes in the scene. Each line of metadata follows the convention of `"{class_name} {mesh_name} {pixel_value}"`. For example if the pixel value of `"keyboard-grey_old"` is 165 in the instance image, its written as `"keyboard keyboard-grey_old 165"`. With this, one can convert the instance segmentation image into a class segmentation image. Note that this metadata shares the same mapping value per scene, for example `"room room-office 240"` is in the metadata from both `"scene01_full_00"` and `"scene01_reduced_00"`, while as `"keyboard-grey_old"` is not present in `"scene01_reduced_00"`, the metadata does not contain `"keyboard keyboard-grey_old 165"` (Fig. 17).

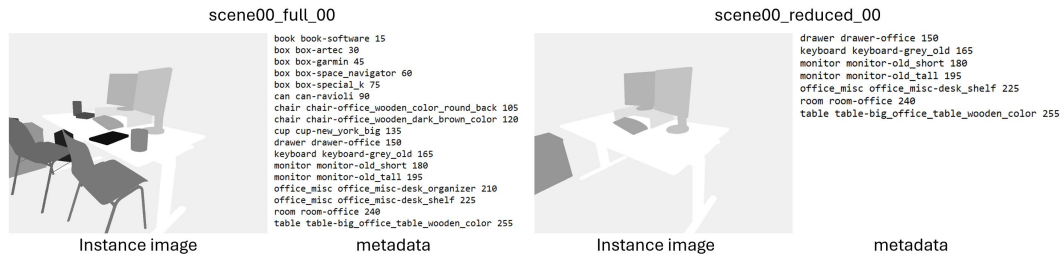



Figure 17: **Example of Instance Segmentation and Metadata Mapping for a Full and Reduced Scene.**

The `"pol"` folder contains polarimetric images obtained from the Lucid Phoenix polarization camera. Polarimetric images are stored as 4 RGB images with 4 different polarization angles (0,45,90,135 degree) arranged in a clockwise manner starting from the left top side. All 4 images are un-distorted so that no distortion coefficient is needed and intrinsic matrix is saved in `"intrinsics.txt"`.

The `"rgb"` folder contains the RGB image sequence. The RGB images are obtained by averaging the 4 polarization images and share the same intrinsics as `"intrinsics.txt"`.

The `"sparse"` folder contains the COLMAP [4, 5] output that is obtained by running COLMAP with the ground truth camera pose. We fix the camera poses to keep the scale unchanged. We specifically provide this folder to make our dataset compatible with 3D Gaussian Splatting [6] variants.

The `"video.avi"` is an example video that serves as qualitative evaluation of the camera trajectory as well as annotation quality. The video contains the visualization of depth maps and depth error and the instance image overlaid on the RGB image as shown in Fig. 18.

12

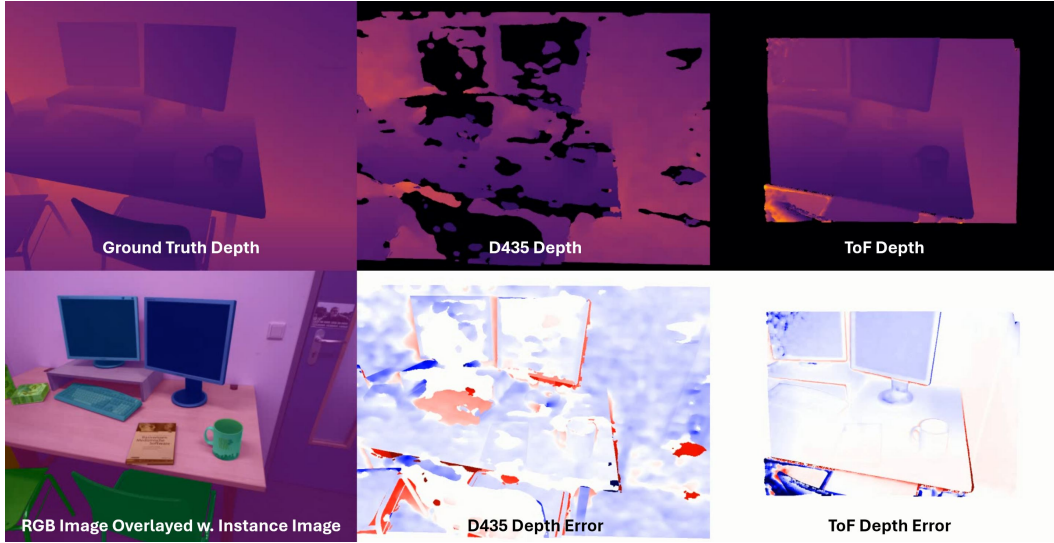



Figure 18: **Example of the** `video.avi` layout.

4.2 Human Reconstruction Dataset

The `Human Reconstruction Dataset` shares a similar file structure as the `Indoor Reconstruction and SLAM Dataset` and the `Object Removal and Scene Editing Dataset`. There a are few differences in the folder arrangement as well as file naming convention as 4 images are captured per human posture. See Fig. 19 for an overview.

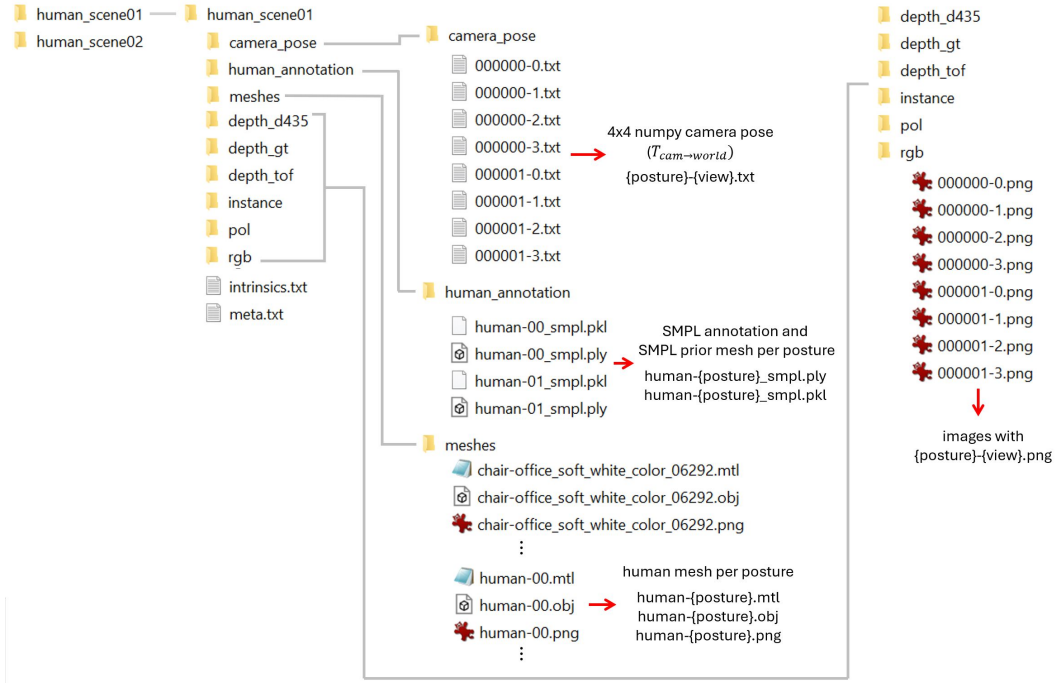



Figure 19: **File Structure of the Human Reconstruction Dataset.**

13

The "meshes" folder contains the meshes of the scene as in the previous dataset setup. However, the folder contains all human meshes from different postures as well. We capture 45 postures for "human_scene_01" and 37 postures for "human_scene_02". Each human posture is scanned with the hand-held scanner in a high resolution and water-tight manner like the other objects meshes and we save the mesh with the naming convention "human-{posture}.obj".

The "human_annotation" folder contains SMPL [7] parameter annotations for each human posture. The *.pkl" files contain SMPL annotations as a python pickle file and the *.ply" files are the SMPL vertices / mesh generated from the pickle file. Both files follow the convention of "human-{posture}_smpl" with extension of *.pkl" and *.ply". The *.pkl"'s SMPL annotation follows the same convention as in [8, 9] that contains a 72 channel vector for the "pose" (first 3 channels are for "global_orient", the remaining 69 channels are for the "body_pose"), a 300 channel vector for the shape "betas" and a 3 channel vector for the translation "trans".

The "camera_pose" folder contains the camera pose for each image frame in the 4x4 matrix format as in the reconstruction and SLAM dataset. However, as the human dataset captures 4 multi-view images on single human posture, it contains 4 poses per posture. Camera poses files are saved with the convention "{posture}-{view}.txt".

All image files including depths, RGBs, instances are saved the same way as in the Reconstruction and SLAM dataset but saved with the same convention as for the camera poses due to the 4 multi-view images captured per human posture. All images follow the same convention of "{posture}-{view}.png".

While "meta.txt" has the same mapping information as in the previous dataset setup, plus the humans with the convention of "human 00 {pixel_value}". For the humans no mesh name is provided in the middle as there are multiple human meshes per scene but all share the same pixel value in the instance images throughout the same scene.

4.3 6D Pose Estimation Dataset

The contents in the "6D Pose Estimation Dataset" share the same principle as in the Indoor Reconstruction and SLAM Dataset and Object Removal and Scene Editing Dataset for the image folders ("depth_d435", "depth_gt", "depth_tof", "depth_instance", "pol", "rgb"), "intrinsics.txt" and "meta.txt" files, while it contains additional folders, such as "pose_meshes_canonical" and "labels" for the 6D pose annotation. An overview of the file structure is shown in Fig. 20.

The "pose_meshes_canonical" folder contains the meshes of the target objects for the pose estimation in canonical orientation in contrast to the meshes in "meshes" in the other dataset format that are defined in world coordinates. The canonical meshes are self-centered (object center is in the center of it's bounding box) and follow specific orientation convention per class or category (Fig. 21, (a)), while the meshes from the previous dataset are centered and oriented according to their layout in the scene (Fig. 21, (b)). Although they have differences in their orientation, we follow the same naming convention as "{class_name}-{instance_name}.obj".

The "labels" folder contains the 6D pose annotation information as a pickle file with the convention of "{frame}_label.pkl". We provide the pickle file in a similar format as the NOCS [10] dataset's labels. Each pickle file contains a python dictionary of the following keys : "model_list", "instance_ids", "class_ids", "scales", "rotations", "translations", "bboxes", "gt_scales".

1. "model_list" is a python list containing a tuple of each object's name and pixel value in the instance image, such as [({obj_1_name}, {instance_value_1}),({obj_2_name}, {instance_value_2}) ...].

2. "instance_ids" is a python list containing the instance value per object in the same order as "model_list", such as [{instance_value_1}, {instance_value_2} ...].

3. "class_ids" is a python list containing the mapping between the class name and a predefined class id in the same order, such as [{class_id_1}, {class_id_2} ...].

4. "scales" is a python list containing the diagonal length of each object in the same order, such as [{diag_length_1}, {diag_length_2} ...].

14

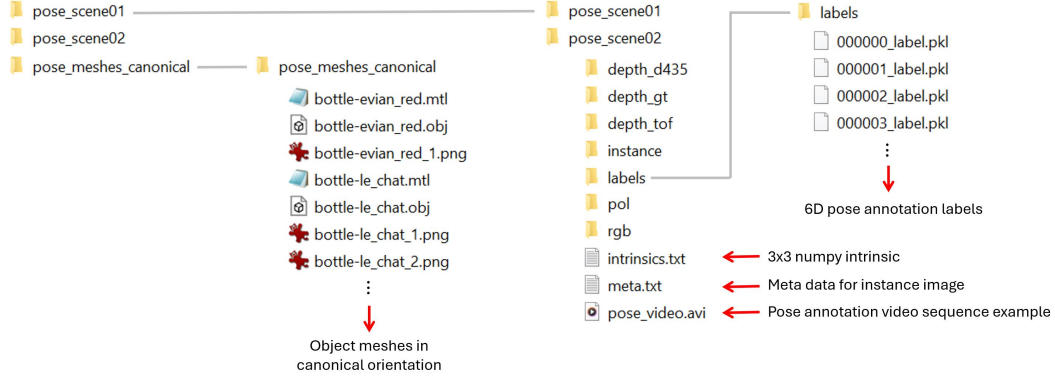



Figure 20: **File Structure of the 6D Pose Estimation Dataset.**

(a) Bottle in a Canonical Orientation   (b) Bottle in a Scene Orientation   (c) Example of "pose_video.avi"

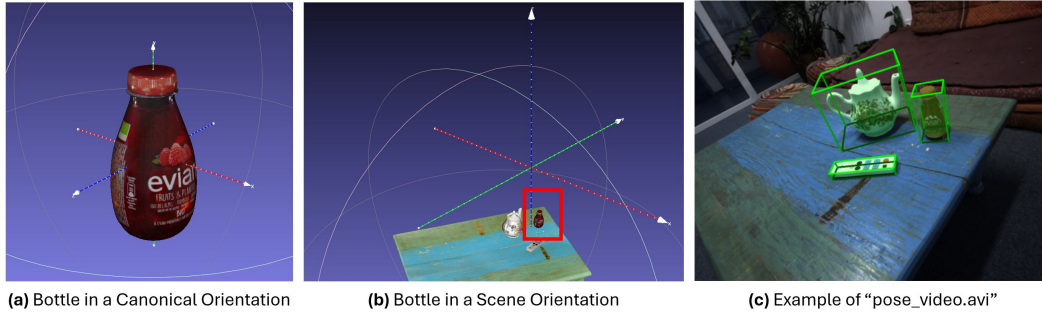



Figure 21: **Example of Meshes and** `"pose_video.avi"`. The object in the canonical orientation is self centered and oriented according to a specific convention (i.e. standing with y-axis), while the object in the scene orientation is oriented according to the scene's layout. The meshes are visualized in Meshlab [11].

5. `"rotation"` is a numpy array of shape $(n\_objects, 3, 3)$ that is an array of rotation matrices for each object stacked along the first dimension in the same order.

6. `"translation"` is a numpy array of shape $(n\_objects, 3)$ that is an array of translation vectors for each object stacked along the first dimension in the same order.

7. `"bboxes"` is a numpy array of shape $(n\_objects, 4)$ that is an array of 2D bounding box annotations in the format `[{top_left_y},{top_left_x},{bottom_right_y},{bottom_right_x}]` stacked along the first dimension in the same order.

8. `"gt_scales"` is a numpy array of shape $(n\_objects, 3)$ that is an array with the mesh size in the format `[{size_x},{size_y},{size_z}]` stacked along the first dimension in the same order.

The `"pose_video.avi"` file contains the visualization of the pose annotation as 3D bounding boxes and rendered object masks on the RGB video for a qualitative evaluation of the pose annotation accuracy as well as camera trajectory coverage (Fig. 21).

15

5 Full Benchmarks

In this section, we show the benchmark results of NVS and SLAM for each scene separately.

5.1 Novel View Synthesis

In this study, the poor performance of NeRFacto [12] in both RGB and depth quality demonstrates the limitations in the model's ability to learn fine details and accurate geometry. In contrast, Gaussian-Splatting [6], Mip-Splatting [13], and Zip-NeRF [14] demonstrate superior performance in RGB image reconstruction. These methods prioritize visual fidelity, resulting in high-quality image outputs. However, the lack of geometric constraints in these models means that while they excel in producing visually appealing images, they fall short in accurately capturing and reconstructing depth information. Focusing on depth estimation, Depth-Facto [12] (AS) and Depth-Facto [12] (ToF) stand out for their exemplary performance in depth estimation metrics. These methods leverage the integration of multi-modal sensor data, which acts as an additional constraint, enhancing their ability to accurately predict depth. The utilization of both AS and ToF sensor data demonstrates the significant advantage of incorporating diverse and complementary data sources, leading to improved depth estimation performance.

Table 1: **NVS Benchmark for scene01**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	22.346	0.801	0.258	0.814	0.414	0.529	0.564	0.728	0.807
Depth-Facto [12] (AS)	25.313	0.828	0.244	0.224	0.086	0.068	0.939	0.956	0.964
Depth-Facto [12] (ToF)	25.267	0.827	0.242	0.300	0.090	0.099	0.908	0.928	0.941
Zip-NeRF [14]	29.548	0.832	0.142	0.322	0.177	0.075	0.647	0.955	0.972
Gaussian-Splatting [6]	28.282	0.848	0.279	0.322	0.133	0.068	0.777	0.904	0.952
Mip-Splatting [13]	28.384	0.848	0.278	0.309	0.122	0.061	0.803	0.925	0.956

Table 2: **NVS Benchmark for scene02**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	24.011	0.808	0.300	1.243	0.528	0.951	0.493	0.635	0.763
Depth-Facto [12] (AS)	26.062	0.835	0.283	0.228	0.092	0.027	0.981	0.995	0.999
Depth-Facto [12] (ToF)	26.251	0.838	0.275	0.739	0.192	0.372	0.855	0.870	0.898
Zip-NeRF [14]	32.278	0.812	0.190	0.374	0.173	0.074	0.640	0.941	0.994
Gaussian-Splatting [6]	29.616	0.842	0.322	0.483	0.195	0.144	0.683	0.852	0.899
Mip-Splatting [13]	29.230	0.842	0.323	0.531	0.218	0.164	0.610	0.824	0.884

Table 3: **NVS Benchmark for scene03**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	20.823	0.808	0.277	2.093	1.482	3.664	0.129	0.224	0.312
Depth-Facto [12] (AS)	24.227	0.869	0.207	0.500	0.199	0.341	0.915	0.939	0.956
Depth-Facto [12] (ToF)	24.524	0.875	0.236	0.960	0.366	0.816	0.777	0.805	0.827
Zip-NeRF [14]	33.581	0.905	0.117	0.368	0.187	0.086	0.616	0.910	0.969
Gaussian-Splatting [6]	28.928	0.920	0.274	0.531	0.253	0.173	0.452	0.788	0.902
Mip-Splatting [13]	29.304	0.920	0.273	0.527	0.257	0.170	0.444	0.760	0.903

16

Table 4: **NVS Benchmark for scene04, sequence00**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	21.880	0.775	0.284	1.102	0.485	0.740	0.431	0.655	0.815
Depth-Facto [12] (AS)	22.803	0.791	0.247	0.262	0.088	0.036	0.950	0.979	0.993
Depth-Facto [12] (ToF)	23.060	0.800	0.246	0.306	0.062	0.055	0.932	0.958	0.977
Zip-NeRF [14]	24.267	0.783	0.232	0.602	0.244	0.208	0.510	0.825	0.914
Gaussian-Splatting [6]	20.037	0.758	0.379	1.002	0.393	0.494	0.321	0.519	0.653
Mip-Splatting [13]	20.605	0.766	0.370	1.032	0.412	0.508	0.271	0.485	0.615

Table 5: **NVS Benchmark for scene04, sequence01**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	23.323	0.771	0.324	1.216	0.723	1.089	0.269	0.428	0.662
Depth-Facto [12] (AS)	25.453	0.803	0.292	0.268	0.084	0.062	0.951	0.962	0.982
Depth-Facto [12] (ToF)	25.370	0.804	0.288	0.284	0.067	0.069	0.933	0.949	0.967
Zip-NeRF [14]	25.046	0.765	0.344	0.781	0.441	0.498	0.352	0.546	0.610
Gaussian-Splatting [6]	23.207	0.784	0.374	0.691	0.340	0.303	0.374	0.589	0.714
Mip-Splatting [13]	23.008	0.782	0.378	0.711	0.359	0.327	0.356	0.559	0.676

Table 6: **NVS Benchmark for scene05**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	20.897	0.755	0.371	1.175	0.672	1.025	0.340	0.500	0.688
Depth-Facto [12] (AS)	23.172	0.786	0.360	0.141	0.059	0.016	0.975	0.984	0.993
Depth-Facto [12] (ToF)	23.038	0.785	0.357	0.202	0.058	0.057	0.941	0.960	0.971
Zip-NeRF [14]	24.206	0.770	0.364	0.633	0.348	0.325	0.410	0.654	0.735
Gaussian-Splatting [6]	25.627	0.817	0.345	0.482	0.224	0.160	0.590	0.791	0.879
Mip-Splatting [13]	25.177	0.814	0.345	0.472	0.222	0.158	0.604	0.801	0.880

Table 7: **NVS Benchmark for scene06**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	23.050	0.809	0.262	0.921	0.420	0.663	0.599	0.789	0.864
Depth-Facto [12] (AS)	24.118	0.813	0.269	0.126	0.035	0.008	0.988	0.997	0.999
Depth-Facto [12] (ToF)	24.106	0.812	0.267	0.098	0.019	0.008	0.988	0.994	0.997
Zip-NeRF [14]	26.724	0.818	0.224	0.403	0.203	0.116	0.623	0.890	0.937
Gaussian-Splatting [6]	26.992	0.833	0.269	0.386	0.141	0.081	0.793	0.948	0.979
Mip-Splatting [13]	26.992	0.833	0.269	0.386	0.141	0.081	0.793	0.948	0.979

Table 8: **NVS Benchmark for scene07**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	22.820	0.672	0.528	0.954	0.401	0.584	0.567	0.745	0.855
Depth-Facto [12] (AS)	23.560	0.677	0.514	0.149	0.055	0.014	0.991	0.995	0.997
Depth-Facto [12] (ToF)	23.530	0.679	0.517	0.166	0.028	0.025	0.979	0.987	0.992
Zip-NeRF [14]	23.559	0.646	0.459	0.737	0.376	0.379	0.273	0.626	0.748
Gaussian-Splatting [6]	24.457	0.693	0.417	0.521	0.210	0.148	0.588	0.834	0.916
Mip-Splatting [13]	24.258	0.693	0.416	0.530	0.212	0.153	0.592	0.829	0.912

17

Table 9: **NVS Benchmark for scene08**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	23.803	0.619	0.535	0.908	0.430	0.567	0.505	0.710	0.855
Depth-Facto [12] (AS)	24.849	0.629	0.517	0.121	0.048	0.009	0.988	0.995	0.999
Depth-Facto [12] (ToF)	24.913	0.632	0.513	0.097	0.023	0.008	0.983	0.992	0.998
Zip-NeRF [14]	26.584	0.629	0.367	0.304	0.153	0.060	0.735	0.971	0.983
Gaussian-Splatting [6]	25.140	0.639	0.416	0.389	0.153	0.084	0.720	0.908	0.961
Mip-Splatting [13]	25.214	0.640	0.415	0.385	0.149	0.082	0.732	0.907	0.960

Table 10: **NVS Benchmark for scene09**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	22.594	0.601	0.583	1.067	0.616	0.923	0.425	0.596	0.715
Depth-Facto [12] (AS)	23.273	0.610	0.571	0.097	0.037	0.006	0.990	0.996	0.999
Depth-Facto [12] (ToF)	23.029	0.607	0.572	0.128	0.030	0.021	0.974	0.983	0.986
Zip-NeRF [14]	23.734	0.603	0.433	0.484	0.270	0.175	0.406	0.766	0.884
Gaussian-Splatting [6]	24.038	0.626	0.427	0.364	0.160	0.084	0.709	0.898	0.958
Mip-Splatting [13]	24.091	0.627	0.428	0.364	0.160	0.086	0.704	0.891	0.954

Table 11: **NVS Benchmark for scene10**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	23.024	0.903	0.191	1.727	0.620	1.839	0.542	0.704	0.771
Depth-Facto [12] (AS)	23.685	0.908	0.203	0.190	0.067	0.017	0.991	0.999	1.000
Depth-Facto [12] (ToF)	23.705	0.909	0.194	0.163	0.030	0.016	0.980	0.993	0.998
Zip-NeRF [14]	36.385	0.942	0.119	0.343	0.137	0.052	0.769	0.993	1.000
Gaussian-Splatting [6]	26.637	0.928	0.231	0.450	0.143	0.099	0.770	0.923	0.963
Mip-Splatting [13]	26.648	0.929	0.229	0.431	0.138	0.091	0.774	0.928	0.968

Table 12: **NVS Benchmark for scene11**

Evaluation	RGB			Depth					
Method	PSNR↑	SSIM↑	LPIPS↓	RSME↓	Abs Rel↓	Sqr Rel↓	$<1.25^1$ ↑	$<1.25^2$ ↑	$<1.25^3$ ↑
NeRFacto [12]	23.166	0.829	0.212	1.716	0.950	2.207	0.287	0.450	0.601
Depth-Facto [12] (AS)	27.510	0.889	0.178	0.330	0.122	0.102	0.958	0.977	0.983
Depth-Facto [12] (ToF)	27.695	0.886	0.174	0.589	0.151	0.246	0.858	0.894	0.928
Zip-NeRF [14]	33.866	0.894	0.121	0.563	0.242	0.217	0.574	0.821	0.895
Gaussian-Splatting [6]	28.360	0.928	0.199	0.691	0.273	0.249	0.474	0.715	0.828
Mip-Splatting [13]	28.299	0.929	0.200	0.715	0.289	0.266	0.439	0.669	0.804

18

5.2 SLAM

This benchmark compares several SLAM methods: NICE-SLAM [15], CO-SLAM [16], and Gaussian-SLAM [17], evaluated using Ground Truth (GT) data, AS (Active Stereo) sensor data, and TOF (Time of Flight) sensor data, highlighting the significant impact of sensor quality on SLAM performance.

NICE-SLAM achieves high-quality scene reconstruction by optimizing a hierarchical representation using pre-trained geometric priors [18]. However, it struggles with hole-filling. CO-SLAM further improves map completion by integrating smoothness and coherence priors into coordinate encodings [19]. On the other hand, Gaussian-SLAM efficiently seeds new Gaussians for newly explored areas and optimizes them online by organizing the scene into independently optimized sub-maps.

When using ground truth data, Gaussian-SLAM excels in both tracking and mapping performance, indicating a strong algorithm highly dependent on input quality. With the AS sensor, all SLAM methods exhibit some performance degradation due to the constant random noise in the depth sensor. Gaussian-SLAM shows substantial deterioration in both tracking and reconstruction performance, especially in scenes with motion blur, resulting in a higher ATE and reduced map completeness. However, CO-SLAM and NICE-SLAM demonstrate better resilience to the noisy depth input from the AS sensor, maintaining relatively higher map accuracy and completeness.

The TOF sensor data further challenges the SLAM methods since the depth quality is influenced by rapid motion. Gaussian-SLAM struggles significantly, even failing in `scene07` due to unreliable photometric and geometric consistency. Its performance drops dramatically across all metrics. Conversely, leveraging hybrid scene representations, NICE-SLAM and CO-SLAM show better robustness towards the noisy depth input, with CO-SLAM maintaining higher map completion rates.

Table 13: **SLAM Benchmarks for scene01.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	1.22	1.47	2.51	88.97
	AS	7.39	10.18	4.69	66.18
	ToF	7.39	17.01	7.03	65.34
CO-SLAM [16]	GT	3.22	1.48	1.46	96.06
	AS	9.18	6.29	3.40	78.19
	ToF	7.61	9.54	4.54	77.12
Gaussian-SLAM [17]	GT	0.86	0.65	0.76	99.64
	AS	9.18	21.53	3.73	74.62
	ToF	4.96	21.92	3.73	74.42

Table 14: **SLAM Benchmarks for scene02.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	7.33	2.93	4.25	78.41
	AS	18.14	15.31	7.91	43.72
	ToF	5.15	18.20	6.84	69.02
CO-SLAM [16]	GT	3.50	1.49	1.54	97.04
	AS	13.27	27.33	13.95	36.21
	ToF	5.62	37.77	4.30	80.71
Gaussian-SLAM [17]	GT	1.83	1.15	1.16	99.06
	AS	17.50	25.75	10.50	33.48
	ToF	29.81	43.69	7.01	56.81

19

Table 15: **SLAM Benchmarks for scene03.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	1.64	2.58	6.11	84.44
	AS	21.79	60.90	18.71	34.50
	ToF	8.19	31.26	12.70	61.87
CO-SLAM [16]	GT	5.55	2.53	1.21	98.07
	AS	18.32	55.16	13.43	39.49
	ToF	18.73	76.67	6.81	64.79
Gaussian-SLAM [17]	GT	5.35	4.64	1.35	96.98
	AS	13.02	120.85	7.66	49.35
	ToF	13.25	59.84	4.87	68.95

Table 16: **SLAM Benchmarks for scene04, sequence00.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	9.89	3.77	9.83	62.29
	AS	45.89	22.20	17.31	26.03
	ToF	45.70	20.25	35.38	23.51
CO-SLAM [16]	GT	4.48	2.01	1.66	98.27
	AS	14.97	30.16	14.08	28.24
	ToF	13.94	97.31	12.54	31.56
Gaussian-SLAM [17]	GT	2.44	1.41	1.26	99.17
	AS	10.43	22.32	12.12	30.15
	ToF	60.36	33.70	16.93	29.58

Table 17: **SLAM Benchmarks for scene04, sequence01.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	11.24	2.29	5.88	78.57
	AS	14.29	27.84	12.68	38.01
	ToF	14.13	20.95	10.18	48.43
CO-SLAM [16]	GT	8.02	2.21	2.13	94.37
	AS	13.74	67.72	15.30	26.15
	ToF	13.86	56.85	6.50	55.97
Gaussian-SLAM [17]	GT	2.53	1.51	1.28	99.68
	AS	35.27	24.03	9.92	37.61
	ToF	60.36	33.70	16.93	29.58

Table 18: **SLAM Benchmarks for scene05.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	15.24	9.34	11.66	50.29
	AS	19.65	11.01	8.28	51.53
	ToF	22.48	24.04	15.53	33.80
CO-SLAM [16]	GT	5.64	2.93	2.78	87.64
	AS	16.37	19.19	8.26	49.41
	ToF	25.66	58.58	12.83	51.28
Gaussian-SLAM [17]	GT	17.90	2.92	2.78	87.63
	AS	15.02	19.25	8.27	49.43
	ToF	18.39	58.71	12.77	51.30

20

Table 19: **SLAM Benchmarks for scene06.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	1.52	0.91	6.64	81.58
	AS	76.70	14.34	9.07	53.28
	ToF	114.70	27.12	13.33	40.22
CO-SLAM [16]	GT	3.23	1.10	1.10	99.16
	AS	8.05	12.59	4.21	71.79
	ToF	5.76	106.43	6.92	69.69
Gaussian-SLAM [17]	GT	1.26	0.77	0.79	99.92
	AS	6.24	19.84	4.44	66.16
	ToF	6.26	65.86	7.34	63.17

Table 20: **SLAM Benchmarks for scene07.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	5.01	3.26	7.39	58.74
	AS	24.81	19.59	18.14	28.77
	ToF	5.96	15.60	8.67	54.41
CO-SLAM [16]	GT	5.22	1.74	1.76	97.57
	AS	14.07	21.75	14.83	33.20
	ToF	11.00	46.05	6.86	51.02
Gaussian-SLAM [17]	GT	1.50	1.30	1.29	99.55
	AS	151.90	28.53	14.04	34.61
	ToF	fail	fail	fail	fail

Table 21: **SLAM Benchmarks for scene08.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	2.39	2.08	6.56	81.59
	AS	80.66	14.95	12.94	35.21
	ToF	11.41	11.51	9.99	57.69
CO-SLAM [16]	GT	3.64	1.61	1.68	96.88
	AS	17.86	23.03	12.87	34.23
	ToF	7.25	32.36	5.71	67.28
Gaussian-SLAM [17]	GT	2.25	1.31	1.35	99.22
	AS	19.46	21.68	10.73	36.49
	ToF	82.26	29.92	6.86	59.76

Table 22: **SLAM Benchmarks for scene09.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	3.93	2.44	4.07	80.08
	AS	30.05	18.92	14.53	35.47
	ToF	22.15	24.98	31.79	20.78
CO-SLAM [16]	GT	2.29	1.66	1.69	97.64
	AS	19.39	17.50	10.74	45.87
	ToF	17.84	38.55	11.45	38.38
Gaussian-SLAM [17]	GT	0.75	1.07	1.12	99.53
	AS	4.28	14.46	6.61	53.73
	ToF	10.41	23.96	4.09	78.55

21

Table 23: **SLAM Benchmarks for scene10.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	5.13	1.94	2.63	91.72
	AS	50.65	34.19	20.48	20.74
	ToF	18.95	25.41	12.89	36.88
CO-SLAM [16]	GT	2.28	1.67	1.59	97.65
	AS	242.38	75.08	21.55	22.98
	ToF	381.76	67.49	8.11	63.62
Gaussian-SLAM [17]	GT	16.06	6.66	4.40	72.90
	AS	56.91	77.00	19.55	16.58
	ToF	108.39	54.44	12.57	31.69

Table 24: **SLAM Benchmarks for scene11.**

Evaluation		Tracking	Mapping		
Methods	Depth	ATE↓ [cm]	Acc↓ [cm]	Comp↓ [cm]	Comp Ratio↑ [%]
NICE-SLAM [15]	GT	1.07	1.93	3.37	88.64
	AS	79.10	34.19	29.38	16.50
	ToF	11.86	16.00	8.72	52.62
CO-SLAM [16]	GT	3.06	1.96	1.70	97.92
	AS	51.78	40.79	19.51	17.23
	ToF	42.3	71.10	9.68	40.71
Gaussian-SLAM [17]	GT	7.21	3.66	2.80	85.46
	AS	119.20	60.03	33.32	14.11
	ToF	69.43	46.15	11.18	36.03

22

6 Licenses of Codes Used in Benchmarks

In this section, we show licenses of codes that are used in the benchmark for NVS (Tab. 25) and SLAM (Tab. 26). Note that our dataset uses the **MIT License**.

Table 25: **License Information of the NVS code Used in the Benchmark.**

Methods	NeRF(Depth)-Facto [12]	Zip-NeRF [14]	Gaussian-Spatting [6]	Mip-Splatting [13]
License	Apache-2.0	Apache-2.0	Inria and MPII	Inria and MPII

Table 26: **License Information of the SLAM code Used in the Benchmark.**

Methods	NICE-SLAM [15]	CO-SLAM [16]	Gaussian-SLAM [17]
License	Apache-2.0	Apache-2.0	MIT

References

[1] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12–22.

[2] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[3] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[4] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[6] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[7] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015.

[8] B. L. Bhatnagar, C. Sminchisescu, C. Theobalt, and G. Pons-Moll, "Combining implicit function learning and parametric models for 3d human reconstruction," in *European Conference on Computer Vision (ECCV)*. Springer, aug 2020.

[9] ——, "Loopreg: Self-supervised learning of implicit surface correspondences, pose and shape for 3d human mesh registration," in *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020.

[10] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008.

[12] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja *et al.*, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, 2023, pp. 1–12.

[13] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-splatting: Alias-free 3d gaussian splatting," *arXiv:2311.16493*, 2023.

23

[14] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Zip-nerf: Anti-aliased grid-based neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 697–19 705.

[15] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[16] H. Wang, J. Wang, and L. Agapito, "Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam," in *CVPR*, 2023.

[17] V. Yugay, Y. Li, T. Gevers, and M. R. Oswald, "Gaussian-slam: Photo-realistic dense slam with gaussian splatting," 2023.

[18] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*.   Springer, 2020, pp. 523–540.

[19] T. Müller, B. Mcwilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance sampling," *ACM Transactions on Graphics*, p. 1–19, Oct 2019. [Online]. Available: http://dx.doi.org/10.1145/3341156

24