This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

4D Gaussian Splatting SLAM

Yanyan Li[1,2], Youxu Fang[1], Zunjie Zhu[1†], Kunyi Li[2], Yong Ding[3], Federico Tombari[2,4]
[1]Hangzhou Dianzi University, [2]Technical University of Munich
[3]Zhejiang University, [4]Google, [†]Corresponding author

Project Page: https://github.com/yanyan-li/4DGS-SLAM

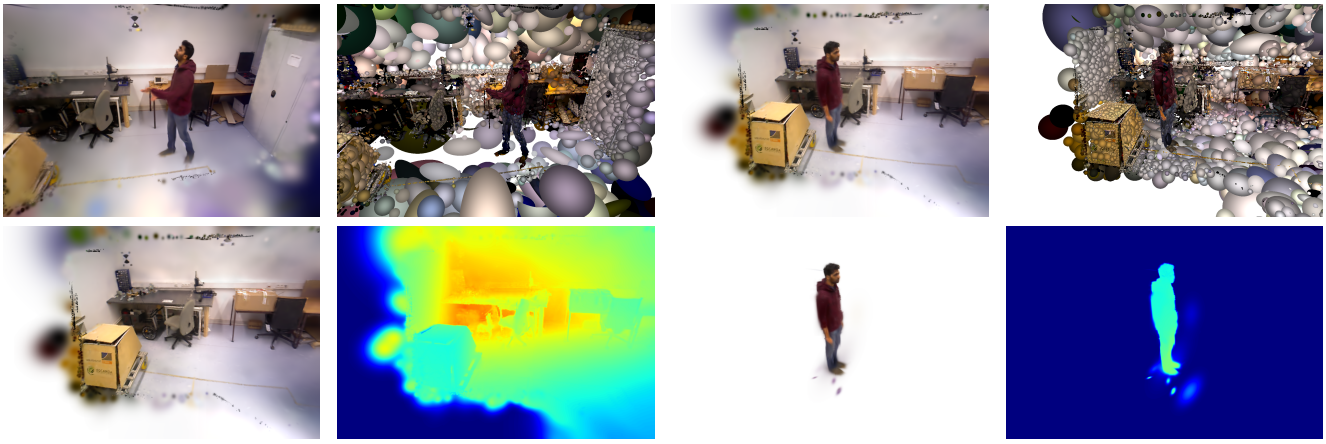

Figure 1. **Example results from the proposed 4D-GS SLAM system**. The top row showcases novel view synthesis and Gaussian visualizations in the BONN balloon (top left) and person_tracking (top right) sequences. The appearance and geometry of static and dynamic scenes are shown in the bottom row, respectively.

Abstract

Simultaneously localizing camera poses and constructing Gaussian radiance fields in dynamic scenes establish a crucial bridge between 2D images and the 4D real world. Instead of removing dynamic objects as distractors and reconstructing only static environments, this paper proposes an efficient architecture that incrementally tracks camera poses and establishes the 4D Gaussian radiance fields in unknown scenarios by using a sequence of RGB-D images. First, by generating motion masks, we obtain static and dynamic priors for each pixel. To eliminate the influence of static scenes and improve the efficiency of learning the motion of dynamic objects, we classify the Gaussian primitives into static and dynamic Gaussian sets, while the sparse control points along with an MLP are utilized to model the transformation fields of the dynamic Gaussians. To more accurately learn the motion of dynamic Gaussians, a novel 2D optical flow map reconstruction algorithm is designed to render optical flows of dynamic objects between neighbor images, which are further used to supervise the 4D Gaussian radiance fields along with traditional photometric and geometric constraints. In experiments, qualitative and quantitative evaluation results show that the proposed method achieves robust tracking and high-quality view synthesis performance in real-world environments.

1. Introduction

Tracking [25, 32], mapping [10, 19], and rendering [22, 36] in dynamic 3D scenes remain a fundamental challenge in computer vision, with important applications in robotics, augmented reality, and autonomous systems. While traditional methods [30, 50, 53] have demonstrated impressive localization and view synthesis capabilities in static environments, the presence of moving objects and diverse lighting conditions in real-world scenarios still significantly limit the performance of current solutions.

3D Gaussian primitives [22, 54] have recently emerged as a powerful representation for novel view synthesis and scene reconstruction, demonstrating efficient performance in training and rendering compared to Neural Radiance Field (NeRF) methods [2, 36]. However, pioneering Gaussian Splatting SLAM algorithms [11, 30] mostly assumed a static working space. Based on photometric and geometric constraints, these methods can incrementally localize

camera poses and optimize Gaussian primitives in unknown scenes. To extend pose estimation capabilities from static scenes to dynamic ones, the most popular strategy [13, 49] is to detect dynamic objects from 2D images and try to remove non-static pixels during the tracking process by leveraging semantic priors [16, 23].

Following a similar dynamic detection strategy, dynamic Gaussian Splatting SLAM [24, 48] systems are proposed to extend the working fields to non-static environments. Based on the support of high-quality dynamic object detection methods [23], the localization accuracy is further improved, also for dynamic Gaussian Splatting SLAM methods. However, after removing the detected dynamic pixel areas, current approaches fall back to reconstructing static Gaussian radiance fields instead of building 4D reconstructions.

To bridge this gap, we introduce a method that simultaneously localizes camera poses and reconstructs 4D Gaussian radiance fields from a sequence of RGB-D images in dynamic scenes. Instead of treating dynamic objects as noise [29] or distractors [38], the proposed approach explicitly models temporal variations of the Gaussian radiance fields, enabling accurate scene representation while maintaining geometric consistency. Our framework incrementally estimates camera poses and updates Gaussian representations in an online manner, ensuring robustness to unknown and highly dynamic environments. By leveraging depth information from RGB-D inputs, we improve geometric accuracy while maintaining efficient computation. Unlike prior work that relies on post-processing or explicit motion segmentation, our method naturally integrates motion cues into the scene representation, allowing for seamless reconstruction without discarding dynamic content. The contributions of our method can be summarized as follows:

- A novel 4D Gaussian Splatting pipeline is proposed to localize camera poses and represent dynamic scenes in Gaussian radiance fields.
- We divide the primitives into static and dynamic Gaussians and introduce sparse control points together with an MLP for modeling the motion of the dynamic Gaussians.
- A novel 2D optical flow rendering algorithm is proposed to improve the performance of 4D Gaussian fields. We estimate the 2D optical flow maps separately from dynamic GS and a pre-trained model, then leverage them as constraints to learn the motion of the dynamic Gaussians.

2. Related Work

Camera Pose Estimation. Camera pose estimation is a fundamental task in communities of computer vision and robotics. Given monocular [7, 33], stereo [8, 32], RGB-D [26, 39], or visual-inertial [3, 35], popular algorithms in the domain of multiple view geometry are proposed to estimate translation and orientation matrices via 2D-2D [12, 28], 2D-3D [14, 52], and 3D-3D [37, 40] strategies. Extended from these fundamental theories, robust and versatile systems [5, 19, 25, 33] are implemented to obtain track cameras and reconstruct unknown environments. There are different focuses between these systems, where the first group [33] of systems pursue accurate localization results while another type [5] of pipelines achieve dense and high-quality 3D reconstructions. With the development of deep neural networks, deep point [6] and line [51] are used in feature matching. RAFT [44] predicts optical flow maps between relative images.

3D Gaussian Splatting and Non-static GS SLAM. 3D Gaussian Splatting (3DGS) [22, 27] is an explicit parametrization for representing 3D unknown scenes, which shows more efficient performance than implicit methods, like NeRF [31] in novel view rendering tasks. For traditional 3DGS methods[21, 30, 50], the application fields mainly focus on static scenes. These approaches have demonstrated strong performance in environments where the scene remains largely unchanged over time, enabling accurate tracking and reconstruction of 3D structures. However, in dynamic scenes, these methods tend to incur significant errors during tracking or reconstruction. For non-static scenes, methods [15, 24, 48] explore strategies to deal with dynamic objects as distractors and establish Gaussian fields for static components after removing dynamic objects. Compared to these non-static Gaussian Splatting methods that assume camera poses are given, non-static GS SLAM methods [24, 48] are incrementally fed by Monocular or RGB-D images to estimate camera poses and reconstruct Gaussian primitives. To achieve the goal, dynamic object instances are masked from 2D images based on semantic detection methods. Furthermore, these removed regions are recovered by multiple views during the optimization process.

Dynamic Gaussian Splatting. Dynamic 3D Gaussian technology enhances the fast rendering capabilities of 3DGS [22], adapting it for dynamic scene reconstruction. In this context, 4D Gaussian splatting [47] (4DGS) presents an innovative approach by combining 3D Gaussians with 4D neural voxels. It introduces a decomposition neural voxel encoding method, inspired by HexPlane [4], to efficiently generate Gaussian features from these 4D neural voxels. To handle temporal variations, a lightweight MLP is applied to predict Gaussian deformations over time. Building on this, the D3DGS framework [1] offers a deformable 3DGS model for dynamic scene representation, where time is conditioned on the 3DGS. This framework transforms the learning process into a canonical space, allowing for the joint training of a purely implicit deformable field with the learnable 3DGS. The result is a time-independent

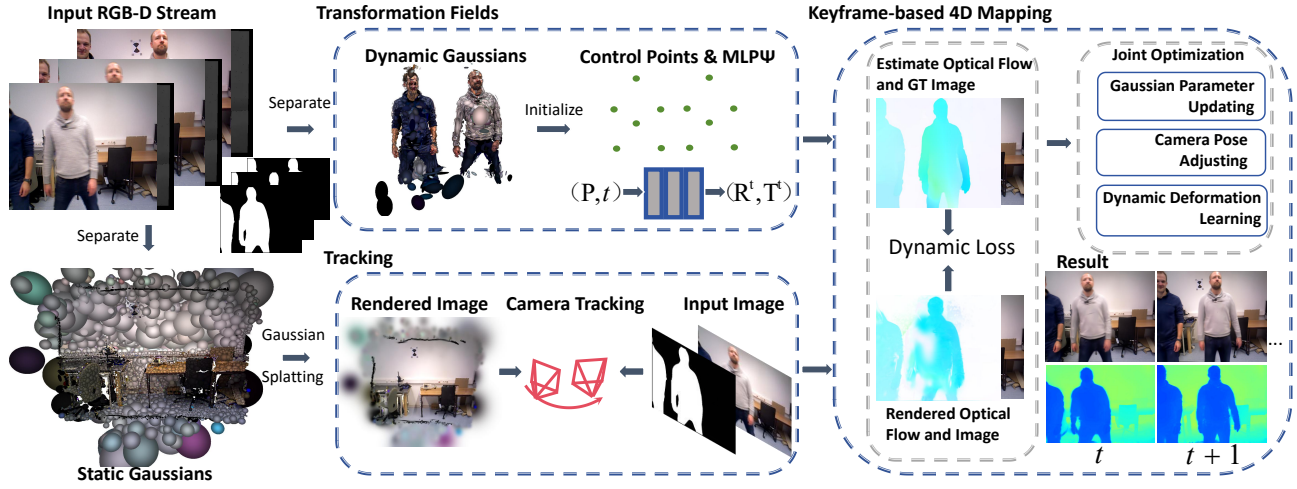

Figure 2. **Architecture of the proposed Gaussian Splatting SLAM.** The inputs to our system are temporally sequential RGB-D image sequences and motion masks. In the initial frame, dynamic and static Gaussians are independently initialized using a motion mask, and sparse control points are established according to the spatial distribution of dynamic Gaussians. The static structure is subsequently employed for camera pose estimation through photometric and geometric constraints. Following keyframe insertion, we co-optimize Gaussian attributes and camera poses while simultaneously estimating temporal motion patterns of dynamic Gaussians.

3DGS that separates motion from geometry. Additionally, 3D Gaussians for Efficient Streaming [43] significantly optimizes the streaming of photo-realistic Free-Viewpoint Videos (FVVs) for dynamic scenes. It achieves this by using a compact Neural Transformation Cache (NTC) to simulate the translation and rotation (transformation fields [17]) of 3D Gaussians. This method reduces the training time and storage space needed for each FVV frame while introducing an adaptive strategy to accommodate new objects in dynamic scenes.

3. Methodology

3.1. Initialization

Similar to GS-based SLAM systems [21, 30, 50], the traditional components of 3D Gaussian ellipsoids, including mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma}$, opacity α, and color $\mathbf{c}$ parameters, are utilized in our representation. But the difference is that we further define a new attribute dy to each Gaussian, which is used to represent whether the Gaussian is a dynamic Gaussian or not. Therefore, the final representation is $\mathcal{G} = [\boldsymbol{\Sigma}\ \boldsymbol{\mu}\ \alpha\ \mathbf{c}\ dy]$.

Following 3D Gaussian Splatting [22], each 3D Gaussian is rasterized into 2D splats, allowing for gradient flow in scene reconstruction and pose estimation. As a result, the rendered color of a pixel, denoted as $C(p)$, can be described by the following equation:

$$C(p) = \sum_{i=1}^{n} \mathbf{c}_i \alpha_i \prod_{j}^{i-1} (1 - \alpha_j) \tag{1}$$

here, $\mathbf{c}$ and α are the color and opacity properties of the Gaussian, respectively.

Additionally, per-pixel depth D(p) and opacity O(p) are rasterized by using alpha-blending:

$$D(p) = \sum_{i=1}^{n} d_i \alpha_i \prod_{j}^{i-1} (1 - \alpha_j) \tag{2}$$

$$O(p) = \sum_{i=1}^{n} \alpha_i \prod_{j}^{i-1} (1 - \alpha_j) \tag{3}$$

where d_i is the distance to the mean $\boldsymbol{\mu}$ of the i^{th} Gaussian along the camera ray.

Instead of assuming that environments are static [21, 30, 50] or removing dynamic objects [24, 48] in Gaussian Splatting optimization, we explore strategies to establish the dynamic deformation network for dynamic Gaussians. To be specific, we use a pre-trained model YoLov9 [46] to obtain the motion mask. For sequences containing dynamic objects that the pre-trained model cannot correctly segment, we generate the motion mask by combining optical flow and the pre-trained model. Based on the detected dynamics, the Gaussians associated with pixels lying on the motion masks are defined as dynamic Gaussians ($\mathcal{G}_{dy}$), while others are initialized as static Gaussians ($\mathcal{G}_{st}$), during the initialization stage.

Inspired by SC-GS [18], we also make use of sparse control points to learn the 6 DoF transformation. However, the difference is that instead of obtaining sparse control points through long-term pre-training, we initialize these points using the motion regions from the input image of the initial frame.

For each control point, we learn a time-varying 6-DoF transformation via an MLP Ψ block. Therefore, the process of querying the transformation fields of each control point P_k at each time step t, which can be denoted as:

$$\Psi(P_k, t) \rightarrow [\mathbf{R}^t, \mathbf{T}^t]. \tag{4}$$

What is more, we derive the dense transformation field of dynamic Gaussians using local interpolation of the transformations of their neighboring control points, employing Linear Blend Skinning (LBS) [42]. Specifically, for each dynamic Gaussian $\mathcal{G}_{dy}$, we use K-Nearest Neighbors (KNN) search to find its K nearest control points $p_k | k \in N_j$ in the canonical space. Then, the interpolation weights for the control points p_k can be computed using a Gaussian Radial Basis Function (RBF). By using the interpolation weights of the neighboring control points and the 6-DoF transformations, we can compute the scale $\mathbf{S}$, rotation $\mathbf{R}$, and positional $\boldsymbol{\mu}$ changes of each dynamic Gaussian $\mathcal{G}_{dy}$.

3.2. Tracking

To avoid interference from the motion of dynamic objects in the input and rendered images on camera tracking, we exclude dynamic Gaussians from the Gaussian splatter rendering during the tracking process. Instead, we optimize the camera pose and exposure parameters using the rendered color and depth maps, which are generated only by static Gaussians. The optimization is performed using L_1 loss between the rendered appearance and depth maps and their observations, where the motion mask $\mathcal{M}$ is used here to remove dynamic objects from the input images to achieve robust camera pose localization performance:

$$L_t = \sum_p \mathcal{M}(\lambda O(p) L_1(C(p)) + (1-\lambda) L_1(D(p))) \tag{5}$$

here, an L1 loss is to supervise both the depth and color renders, and λ is a fixed weight during the optimization process. Note that, for $L_1(D(p))$, we only apply the loss over pixels that $O(p) > 0.95$ and the ground-truth depth $d(p) > 0$. For $L_1(C(p))$, we only apply the loss over pixels where the gradient of the ground-truth color image exceeds a certain threshold σ.

Keyframe Selection. Similar to MonoGS [30], we also maintain a small number of keyframes in the sliding window W, using visibility checks and translation thresholds to select keyframes, removing them if their overlap with the latest keyframe drops below a threshold. However, a new strategy, different from MonoGS [30], is proposed by considering dynamic situations. Specifically, even if the camera movement is small, a new keyframe can also be selected and inserted when we detect the motion mask has a big difference or at least every 5 frames. After adding a keyframe, we initialize new static Gaussians with the static part of the input image pixels from the current frame, followed by the mapping step. However, new dynamic Gaussians will not be added.

3.3. 4D Mapping

Once new static and dynamic scenarios are inserted into the system after the tracking process, we propose a 4D mapping module to optimize the dynamic Gaussian radiance fields.

Optical Flow Map Rendering. As introduced in Equation 5, appearance (RGB) and geometry (depth) rendering constraints are utilized in the tracking process. However, in the 4D mapping section, these traditional single-view supervisions can provide reliable constraints for dynamic scenarios incrementally.

To solve the problem, we are the first 4D Gaussian Splatting SLAM system that provides a novel strategy to render another type of map, the Optical Flow Map, in the 4D mapping module. First of all, to create accurate optical flows between two images, the traditional methods [9] use pixel-based tracking methods. Instead of from the perspective of 2D views and correspondence matching, we migrate the dynamic Gaussians $\mathcal{G}_{dy}$ between the currently selected keyframe and its last keyframe to obtain two corresponding sets of Gaussians, G_t and G_{t-1}. These two sets of Gaussians are projected onto the camera plane of the current keyframe, resulting in two sets of 2D point coordinates $\mathbf{p}_t$ and $\mathbf{p}_{t-1}$. Let the difference between p_t and p_{t-1} be denoted as d_x. Similar to rendering color and depth maps, we can use d_x to render the backward optical flow map $F(p)$ from time t to $t-1$:

$$F(p) = \sum_{i=1}^{n} d_x \alpha_i \prod_{j}^{i-1} (1-\alpha_j). \tag{6}$$

Similarly, we can also render the forward optical flow map from frames I_{t-1} to I_t. The optical flow loss is computed by comparing the forward and backward optical flow maps rendered from the dynamic Gaussians with the forward and backward optical flow maps estimated by RAFT [45] for the real input color images at times $t-1$ and t in the motion mask area, using $L1$ loss, which can be denoted as:

$$\begin{aligned} \mathcal{L}_{flow} = \sum_p \mathcal{M}(L_1(F(p)_{t\rightarrow t-1}, RAFT(p)_{t\rightarrow t-1}) \\ + L_1(F(p)_{t-1\rightarrow t}, RAFT(p)_{t-1\rightarrow t})) \end{aligned} \tag{7}$$

here, $F(p)_{t\rightarrow t-1}$ and $F(p)_{t-1\rightarrow t}$ are the optical flow maps of dynamic Gaussian rendering from time t to t-1 and from time t-1 to t, $RAFT(p)_{t\rightarrow t-1}$ and $RAFT(p)_{t-1\rightarrow t}$ are the optical flow map estimated by RAFT [45] from time t to $t-1$ and time $t-1$ to t.

Method	ballon	ballon2	ps_track	ps_track2	sync	sync2	p_no_box	p_no_box2	p_no_box3	Avg.
RoDyn-SLAM[20]	7.9	11.5	14.5	13.8	**1.3**	1.4	4.9	6.2	10.2	7.9
MonoGS[30]	29.6	22.1	54.5	36.9	68.5	0.56	71.5	10.7	3.6	33.1
Gaussian-SLAM[50]	66.9	32.8	107.2	114.4	111.8	164.8	69.9	53.8	37.9	84.3
SplaTAM[21]	32.9	30.4	77.8	116.7	59.5	66.7	91.9	18.5	17.1	56.8
Ours	**2.4**	**3.7**	**8.9**	**9.4**	2.8	**0.56**	**1.8**	**1.5**	**2.2**	**3.6**

Table 1. **Trajectory errors in ATE [cm]↓ in the BONN sequences.** Results with the best accuracy are highlighted in **bold** font.

Method	Metric	fr3/sit_st	fr3/sit_xyz	fr3/sit_rpy	fr3/walk_st	fr3/walk_xyz	fr3/walk_rpy	Avg.
MonoGS[30]	PSNR[dB] ↑	19.95	23.92	16.99	16.47	14.02	15.12	17.74
	SSIM↑	0.739	0.803	0.572	0.604	0.436	0.497	0.608
	LPIPS↓	0.213	0.182	0.405	0.355	0.581	0.56	0.382
Gaussian-SLAM[50]	PSNR[dB] ↑	18.57	19.22	16.75	14.91	14.67	14.5	16.43
	SSIM↑	0.848	0.796	0.652	0.607	0.483	0.467	0.642
	LPIPS↓	0.291	0.326	0.521	0.489	0.626	0.630	0.480
SplaTAM[21]	PSNR[dB] ↑	24.12	22.07	19.97	16.70	17.03	16.54	19.40
	SSIM↑	**0.915**	**0.879**	**0.799**	0.688	0.650	0.635	0.757
	LPIPS↓	**0.101**	**0.163**	**0.205**	0.287	0.339	0.353	0.241
SC-GS[18]	PSNR[dB] ↑	27.01	21.45	18.93	20.99	**19.89**	16.44	20.78
	SSIM↑	0.900	0.686	0.529	0.762	0.590	0.475	0.657
	LPIPS↓	0.182	0.369	0.512	0.291	0.470	0.554	0.396
Ours	PSNR[dB] ↑	**27.68**	**24.37**	**20.71**	**22.99**	19.83	**19.22**	**22.46**
	SSIM↑	0.892	0.822	0.746	**0.820**	**0.730**	**0.708**	**0.786**
	LPIPS↓	0.116	0.179	0.265	**0.195**	**0.281**	**0.337**	**0.228**

Table 2. **Quantitative results in the TUM RGB-D sequences.** Results with the best accuracy are highlighted in **bold** font.

Joint Optimization. In the mapping process, we use the first three keyframes in W and randomly select five keyframes that overlap with the current frame to reconstruct the currently visible area. Additionally, to prevent forgetting the global map, two keyframes are randomly selected during each iteration. We optimize the Gaussian parameters and the camera poses of the three most recently added keyframes using the photometric $\mathcal{L}_1(C(p))$, geometric $\mathcal{L}_1(D(p))$.

And we also introduce the regularization $\mathcal{L}_{iso}$ loss functions to penalize the stretch of the ellipsoid s_i by its difference to the mean $\tilde{s_i}$:

$$E_{iso} = \Sigma_{i=1}^{|\mathcal{G}|} ||s_i - \tilde{s_i}||_1. \tag{8}$$

Furthermore, we optimize the dynamic deformation network, which includes the MLP layers Ψ and the parameters of control points. To achieve this, we also need to compute the ARAP loss [18] and the optical flow loss for each map keyframe.

Finally, we optimize the relevant parameters by using a weighted sum of these losses, denoted as $L_{mapping}$.

$$\begin{aligned} L_{mapping} = \lambda L_1(C(p)) + (1-\lambda)L_1(D(p)) \\ + \lambda_{flow}\mathcal{L}_{flow} + W_1 arap\_loss \\ + W_2 E_{iso} \end{aligned} \tag{9}$$

here, λ, λ_{flow}, $W1$ and $W2$ are fixed weights during optimization.

Therefore, the two-stage mapping strategy is introduced to optimize the camera poses, exposure parameters, and dynamic deformation network. This strategy can be described in detail as follows:

- In the first stage, we use loss mapping $L_{mapping}$ to optimize only the camera poses and exposure parameters for the first three keyframes in W, as well as the dynamic deformation network, without optimizing the Gaussian parameters. During this stage, the weight of the L1 loss for the color and depth maps in the motion mask region will be doubled.
- In the second stage, we use $L_{mapping}$ to optimize the camera poses and exposure parameters for the first three keyframes in W, the dynamic deformation network, and Gaussian parameters.

Method	Metric	ballon	ballon2	ps_track	ps_track2	sync	sync2	p_no_box	p_no_box2	p_no_box3	Avg.
MonoGS[30]	PSNR[dB] ↑	21.35	20.22	20.53	20.09	22.03	20.55	20.764	19.38	24.81	21.06
	SSIM↑	0.803	0.758	0.779	0.718	0.766	0.841	0.748	0.753	0.857	780
	LPIPS↓	0.316	0.354	0.408	0.426	0.328	0.5210	0.428	0.372	0.243	0.342
Gaussian-SLAM[50]	PSNR[dB] ↑	20.45	18.55	19.60	19.09	21.04	21.35	19.99	20.35	21.22	20.18
	SSIM↑	0.792	0.718	0.744	0.719	0.784	0.837	0.750	0.768	0.814	0.769
	LPIPS↓	0.457	0.480	0.484	0.496	0.402	0.364	0.509	0.493	0.441	0.458
SplaTAM[21]	PSNR[dB] ↑	19.65	17.67	18.30	15.57	19.33	19.67	20.81	21.69	21.41	19.34
	SSIM↑	0.781	0.702	0.670	0.606	0.776	0.730	0.824	0.852	0.873	0.757
	LPIPS↓	0.211	0.280	0.283	0.331	0.227	0.258	0.191	0.165	0.152	0.233
SC-GS[18]	PSNR[dB] ↑	22.3	21.38	-	-	**23.62**	22.74	20.60	21.55	19.24	21.63
	SSIM↑	0.737	0.708	-	-	0.788	0.801	0.688	0.722	0.628	0.724
	LPIPS↓	0.448	0.450	-	-	0.427	0.359	0.515	0.491	0.539	0.461
Ours	PSNR[dB] ↑	**25.90**	**22.71**	**21.78**	**20.65**	23.25	**25.42**	**23.14**	**24.28**	**25.88**	**23.66**
	SSIM↑	**0.874**	**0.838**	**0.832**	**0.820**	**0.812**	**0.892**	**0.845**	**0.873**	**0.886**	**0.852**
	LPIPS↓	**0.234**	**0.264**	**0.289**	**0.294**	**0.250**	**0.169**	**0.239**	**0.224**	**0.207**	**0.241**

Table 3. **Quantitative results in the BONN sequences.** Results with the best accuracy are highlighted in **bold** font. And "-" means that reconstruction failure.

Method	fr3/sit_st	fr3/sit_xyz	fr3/sit_rpy	fr3/walk_st	fr3/walk_xyz	fr3/walk_rpy	Avg.
RoDyn-SLAM[20]	1.5	5.6	5.7	1.7	8.3	8.1	5.1
MonoGS[30]	**0.48**	1.7	6.1	21.9	30.7	34.2	15.8
Gaussian-SLAM[50]	0.72	**1.4**	21.02	91.50	168.1	152.0	72.4
SplaTAM[21]	0.52	1.5	11.8	83.2	134.2	142.3	62.2
Ours	0.58	2.9	**2.6**	**0.52**	**2.1**	**2.6**	**1.8**

Table 4. **Trajectory errors in ATE [cm]↓ in the TUM RGB-D sequences.** Results with the best accuracy are highlighted in **bold** font.

Color Refinement. Finally, we perform 1500 iterations of global optimization. In each iteration, we randomly select 10 frames from all keyframes to optimize the dynamic deformation network and Gaussian parameters. The loss used is

$$\begin{aligned} Loss = 0.2D\text{-}SSIM + 0.8L_1(C(p)) \\ + 0.1L_1(D(p)) + W_1 \text{arap\_loss} + W_2 E_{\text{iso}} \end{aligned} \tag{10}$$

here, W_1 and W_2 are fixed weights.

4. Experiments

4.1. Datasets

We evaluate our method on two real-world public datasets: the TUM RGB-D dataset [41] and the BONN RGB-D Dynamic dataset [34]. Both datasets capture indoor scenes using a handheld camera and provide the ground-truth trajectories.

4.2. Implementation

Our method is implemented in Python using the PyTorch framework, incorporating CUDA code for time-critical rasterization and gradient computation of Gaussian splatting, and we run our SLAM on a desktop with Intel(R) Xeon(R) Silver 4210R and a single NVIDIA GeForce RTX 3090 Ti. Furthermore, we set the weight $W_1 = 1e-4$, $W_2 = 10$, $\lambda = 0.9$, $\lambda_{flow} = 3$, $\sigma = 0.01$, for all evaluations.

For sequences where dynamic objects appear in the middle, such as the sequence *placing_nonobstructing_box* of the BONN dataset, we pre-specify the initial frame for initializing dynamic Gaussians and control points.

4.3. Baselines and Metrics

We primarily compare our method to existing GS-SLAM methods such as SplaTAM [21], Gaussian-SLAM [50], and MonoGS [30], as well as Dynamic Gaussian Splatting methods like SC-GS [18], and the NeRF-SLAM method for dynamic scenes, RoDyn-SLAM [20]. Additionally, for SC-GS, we select one image out of every five frames in the

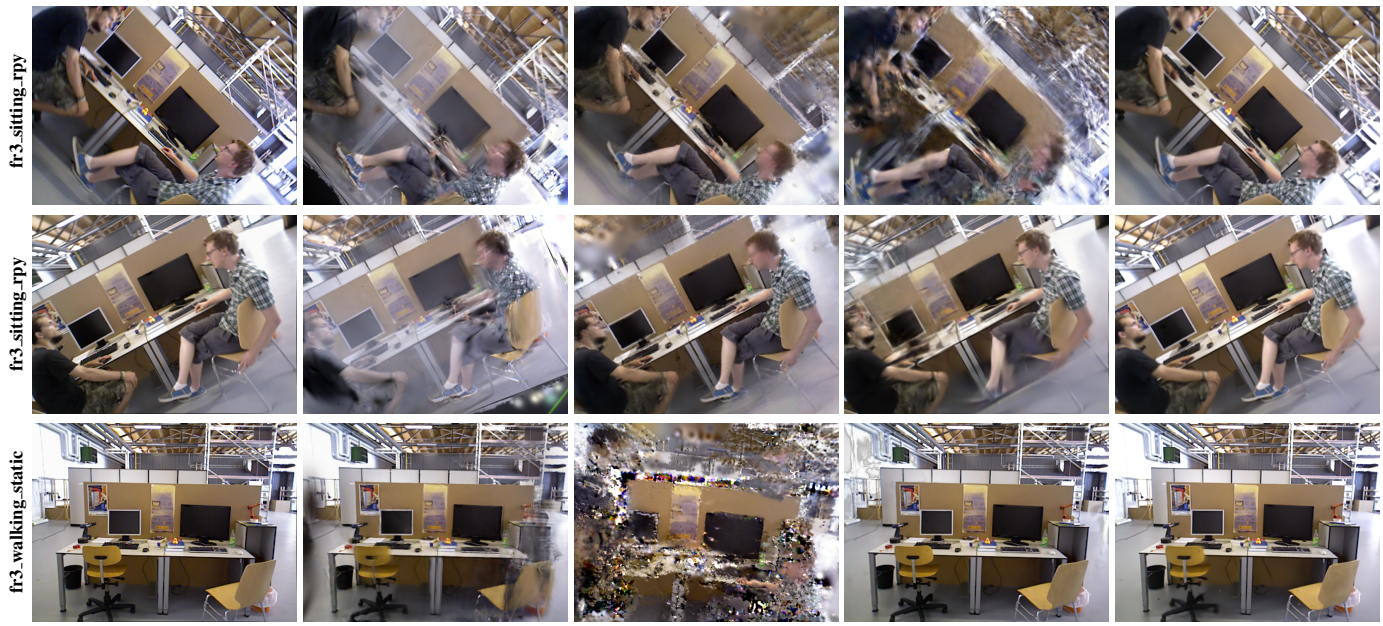

Figure 3. **Visual comparison of the rendering images on the TUM RGB-D dataset.** More results are added to the project page.

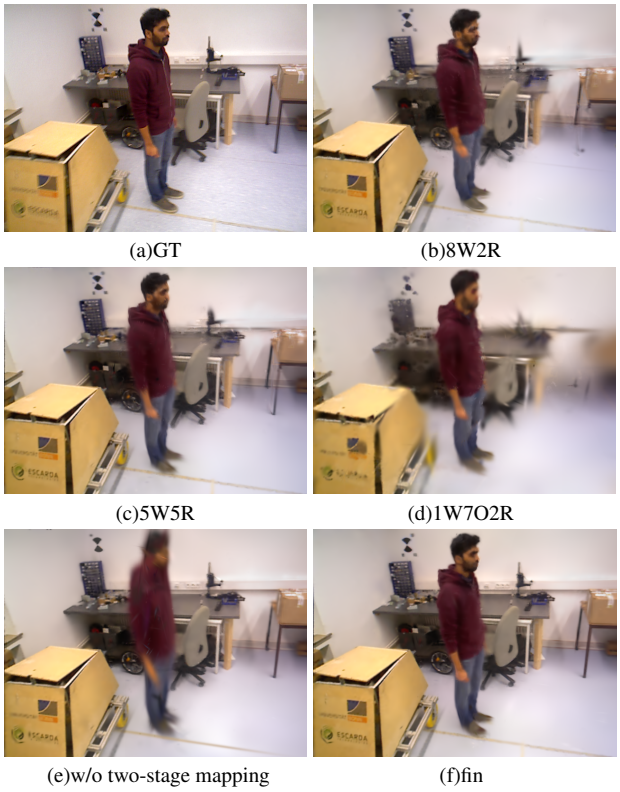

Figure 4. **The comparison of rendering results with different mapping strategies on the BONN RGB-D dynamic dataset**.

dataset as the training dataset, and provide the ground truth camera trajectory and the 3D model obtained by our method for training.

We use standard photometric rendering quality metrics to evaluate the performance of view synthesis, including Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). Given that camera pose estimation performance is crucial for SLAM methods, we also report the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) across all sequences.

4.4. Pose Estimation

Besides rendering performance in appearance, we also evaluate the pose estimation performance of these methods. As shown in Table 4 and 1, the estimated trajectories are compared to the ground truth ones. Thanks to the motion masks and separation of dynamic Gaussians, the proposed method shows robust and accurate camera pose estimation results in high-dynamic scenes compared to these GS-based SLAM methods. Furthermore, our method achieves more accurate results in most of the scenes compared to the NeRF-based dynamic SLAM, RoDyn-SLAM [20].

4.5. Quality of Reconstructed Map

Table 2 and 3 demonstrate the quality of the reconstructed map on the TUM RGB-D [41] and BONN [34] datasets, respectively. We evaluated rendering quality by averaging the differences between the rendered images and the ground truth images across all frames. As shown in Figure 2

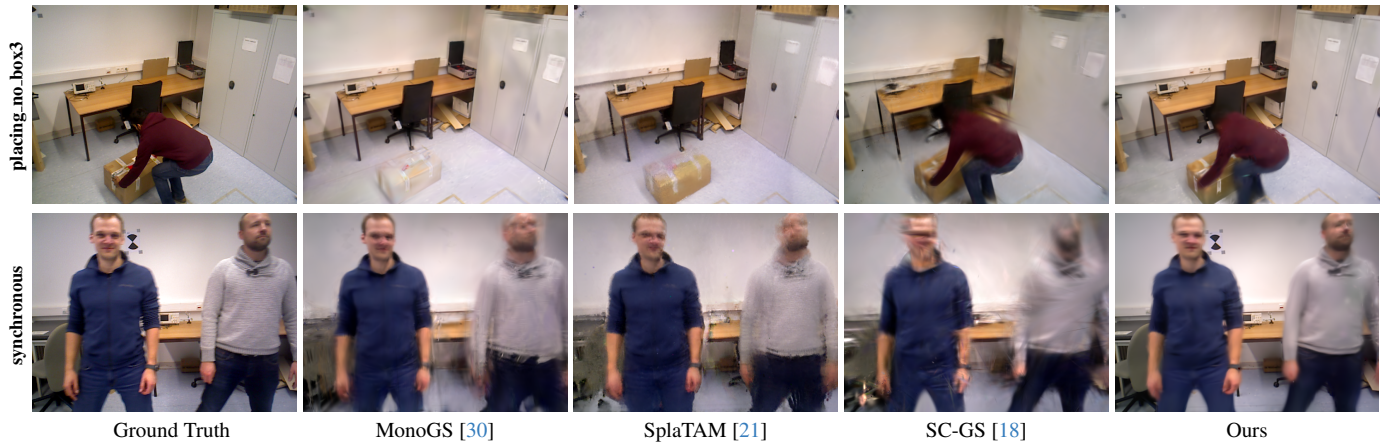

Figure 5. **Visual comparison of the rendering image on the BONN RGB-D dataset.** This is also supported by the quantitative results in Table 3. More qualitative results have been added to the project page.

Optical Flow	Separate Gaussians	syn	syn2
✗	✗	18.37	22.11
✗	✓	22.87	24.84
✓	✗	17.40	21.03
✓	✓	**23.25**	**25.42**

Table 5. Analysis of the impact of Optical Flow Loss and Separate Gaussians on quantitative results (PSNR [dB] ↑) for the *synchronous* and *synchronous2* sequences in the BONN RGB-D dynamic dataset.

and 3, our proposed method achieves better reconstruction than GS-based SLAM and dynamic Gaussian splatting SC-GS [18] in most scenes. Due to the influence of exposure parameters, our method may perform slightly worse on some sequence metrics compared to other methods. However, as shown in Figure 5, our method achieves the best reconstruction of static scenes and dynamic objects. More rendering results are provided in the supplementary material.

4.6. Ablation Study

Mapping Strategy. In Figure 4, we show the impact of different mapping strategies on the final rendering result. Figure 4b represents the results of optimizing the first eight keyframes in the keyframe window and two randomly selected keyframes from all keyframes during the mapping process. Figure 4c represents the results of optimizing the first five keyframes in the keyframe window and five randomly selected keyframes from all keyframes during the mapping process. Figure 4d presents the results of optimizing the first keyframes in the keyframe window, two randomly selected keyframes from all keyframes, and seven randomly chosen keyframes that overlap with the current frame during the mapping process. Figure 4e shows the result of applying the same operation in the first-stage mapping as in the second-stage mapping, the keyframe selection during mapping is the same as in Figure 4f, where Figure 4f is the method we use for mapping, which presents the results of optimizing the first three keyframes in the keyframe window, two randomly selected keyframes from all keyframes, and five randomly chosen keyframes that overlap with the current frame during the mapping process, achieving the best result in both dynamic and static scene reconstruction.

Optical-flow Loss and Separate Gaussians. In Table 5, we ablate two aspects of our system: (1) whether optical flow loss is used during the mapping stage, and (2) whether only the dynamic Gaussian deformation is learned. We do this using *synchronous* and *synchronous2* sequences of the BONN dataset. The results listed in Table 5 demonstrate that the combined use of optical flow loss and dynamic Gaussian separation is effective in scene reconstruction.

5. Conclusion

In this paper, we propose a novel approach for reconstructing dynamic scenes using 4D Gaussian Splatting SLAM. Our method incrementally tracks camera poses and reconstructs dynamic scenes from a sequence of RGB-D images in unknown environments. By leveraging the power of dynamic and static Gaussian segmentation and optical flow, our approach not only localizes the camera and reconstructs the static environment but also effectively maps dynamic objects. We demonstrate its effectiveness in achieving state-of-the-art results in camera pose estimation and dynamic scene reconstruction.

References

[1] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024. 2

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 1

[3] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 2

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *CVPR*, 2023. 2

[5] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4): 1, 2017. 2

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 2

[7] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014. 2

[8] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1935–1942. IEEE, 2015. 2

[9] David Fleet and Yair Weiss. Optical flow estimation. In *Handbook of mathematical models in computer vision*, pages 237–257. Springer, 2006. 4

[10] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. *arXiv preprint arXiv:2312.07504*, 2023. 1

[11] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam, 2024. 1

[12] Robert M Haralick, Hyonam Joo, Chung-Nan Lee, Xinhua Zhuang, Vinay G Vaidya, and Man Bae Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1426–1446, 1989. 2

[13] Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. Dynamic slam: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129. IEEE, 2020. 2

[14] Joel A Hesch and Stergios I Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390. IEEE, 2011. 2

[15] Chenfeng Hou, Qi Xun Yeo, Mengqi Guo, Yongxin Su, Yanyan Li, and Gim Hee Lee. Mvgsr: Multi-view consistency gaussian splatting for robust surface reconstruction. *arXiv preprint arXiv:2503.08093*, 2025. 2

[16] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4421–4430, 2019. 2

[17] Bingbing Hu, Yanyan Li, Rui Xie, Bo Xu, Haoye Dong, Junfeng Yao, and Gim Hee Lee. Learnable infinite taylor gaussian for dynamic view rendering. *arXiv preprint arXiv:2412.04282*, 2024. 3

[18] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023. 3, 5, 6, 7, 8

[19] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 1, 2

[20] Haochen Jiang, Yueming Xu, Kejie Li, Jianfeng Feng, and Li Zhang. Rodyn-slam: Robust dynamic dense rgb-d slam with neural radiance fields. *IEEE Robotics and Automation Letters*, 2024. 5, 6, 7

[21] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track and map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 3, 5, 6, 7, 8

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 3

[23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 2

[24] Mangyu Kong, Jaewon Lee, Seongwon Lee, and Euntai Kim. Dgs-slam: Gaussian splatting slam in dynamic environment. *arXiv preprint arXiv:2411.10722*, 2024. 2, 3

[25] Yanyan Li, Nikolas Brasch, Yida Wang, Nassir Navab, and Federico Tombari. Structure-slam: Low-drift monocular slam in indoor environments. *IEEE Robotics and Automation Letters*, 5(4):6583–6590, 2020. 1, 2

[26] Yanyan Li, Raza Yunus, Nikolas Brasch, Nassir Navab, and Federico Tombari. Rgb-d slam with structural regularities. In *2021 IEEE international conference on Robotics and automation (ICRA)*, pages 11581–11587. IEEE, 2021. 2

[27] Yanyan Li, Chenyu Lyu, Yan Di, Guangyao Zhai, Gim Hee Lee, and Federico Tombari. Geogaussian: Geometry-aware gaussian splatting for scene rendering. In *European Conference on Computer Vision*, pages 441–457. Springer, 2024. 2

[28] Q-T Luong and Olivier D Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices. *International Journal of computer vision*, 22: 261–289, 1997. 2

[29] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024. 2

[30] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2, 3, 4, 5, 6, 7, 8

[31] 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. 2

[32] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 1, 2

[33] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 2

[34] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. 2019. 6, 7

[35] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. 2

[36] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3437–3444. IEEE, 2023. 1

[37] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 2

[38] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20626–20636, 2023. 2

[39] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019. 2

[40] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, page 435. Seattle, WA, 2009. 2

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012. 6, 7

[42] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80–es, 2007. 4

[43] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20675–20685, 2024. 3

[44] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. 2

[45] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020. 4

[46] Chien-Yao Wang and Hong-Yuan Mark Liao. YOLOv9: Learning what you want to learn using programmable gradient information. 2024. 3

[47] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024. 2

[48] Yueming Xu, Haochen Jiang, Zhongyang Xiao, Jianfeng Feng, and Li Zhang. Dg-slam: Robust dynamic gaussian splatting slam with hybrid pose optimization. *Advances in Neural Information Processing Systems*, 37:51577–51596, 2025. 2, 3

[49] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1168–1174. IEEE, 2018. 2

[50] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2023. 1, 2, 3, 5, 6

[51] Ziheng Zhang, Zhengxin Li, Ning Bi, Jia Zheng, Jinlei Wang, Kun Huang, Weixin Luo, Yanyu Xu, and Shenghua Gao. Ppgnet: Learning point-pair graph for line segment detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7105–7114, 2019. 2

[52] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2351, 2013. 2

[53] Zunjie Zhu, Youxu Fang, Xin Li, Chengang Yan, Feng Xu, Chau Yuen, and Yanyan Li. Robust gaussian splatting slam by leveraging loop closure. *arXiv preprint arXiv:2409.20111*, 2024. 1

[54] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 1