DynaGSLAM: Real-Time Gaussian-Splatting SLAM for Online Rendering, Tracking, Motion Predictions of Moving Objects in Dynamic Scenes

Runfa Blark Li[*†] Mahdi Shaghaghi[†] Keito Suzuki[*] Xinshuang Liu[*] Varun Moparthi[*]
Bang Du[*] Walker Curtis[†] Martin Renschler[†] Ki Myung Brian Lee[*]
Nikolay Atanasov[*] Truong Nguyen[*]

[*]UC San Diego
[†]Qualcomm XR Advanced Technology

{runfa, k3suzuki, xil235, vmoparthi, b7du, kmblee, natanaso, tqn001}@ucsd.edu
{shaghagh, wcurtis}@qti.qualcomm.com

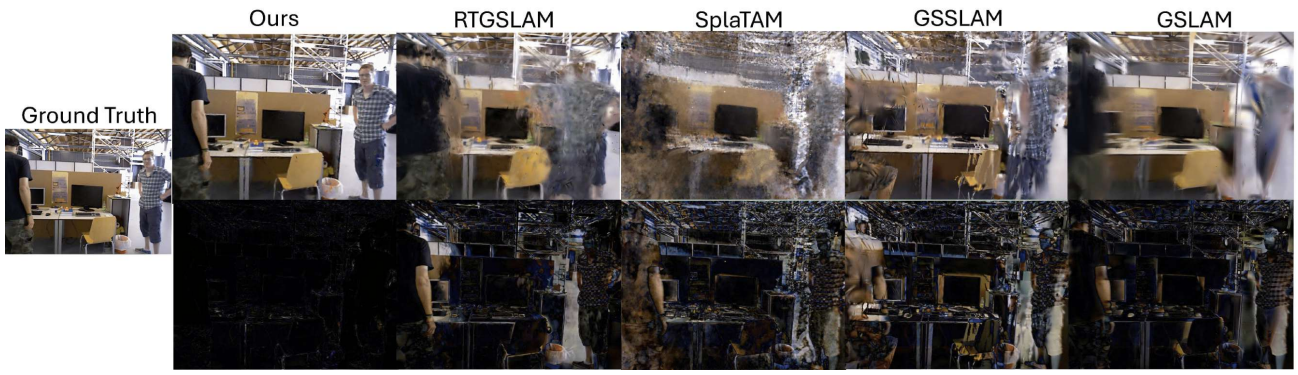

Figure 1. Based on our proposed dynamic GS mapping algorithm, we build **DynaGSLAM**, the first real-time Gaussian-Splatting (GS) based SLAM for online high-quality rendering of dynamic objects in dynamic scenes. With the online RGBD frames, DynaGSLAM enables us to track(interpolate)/predict(extrapolate) the continuous object motions in the past/future, and estimates localization. This figure shows the rendering of GS mapping on TUM dataset [64] [1]with moving people. First row: RGB rendering. Second row: Absolute error between the rendering and the ground truth.

Abstract

Simultaneous Localization and Mapping (SLAM) is one of the most important environment-perception and navigation algorithms for computer vision, robotics, and autonomous cars/drones. Hence, high quality and fast mapping becomes a fundamental problem. With the advent of 3D Gaussian Splatting (3DGS) as an explicit representation with excellent rendering quality and speed, state-of-the-art (SOTA) works introduce GS to SLAM. Compared to classical pointcloud-SLAM, GS-SLAM generates photometric information by learning from input camera views and synthesizing unseen views with high-quality textures. However, these GS-SLAM fail when moving objects occupy the scene that violates the static assumption of bundle adjustment. The failed updates of moving GS affects the static GS and contaminates the full map over the video sequence. Although some efforts have been made by concurrent works to consider moving objects for GS-SLAM, they simply detect and remove the moving regions from GS rendering ("anti" dynamic GS-SLAM), where only the static background could benefit from GS. To this end, we propose the first real-time GS-SLAM, "DynaGSLAM", that achieves high-quality online GS rendering, tracking, motion predictions of moving objects in dynamic scenes while jointly estimating accurate ego motion. Our DynaGSLAM outperforms SOTA static & "Anti" dynamic GS-SLAM on three dynamic real datasets, while keeping speed and memory efficiency in practice. https://blarklee.github.io/dynagslam/

[1]TUM RGB-D SLAM dataset and benchmark is licensed under Creative Commons 4.0 Attribution License (CC BY 4.0).

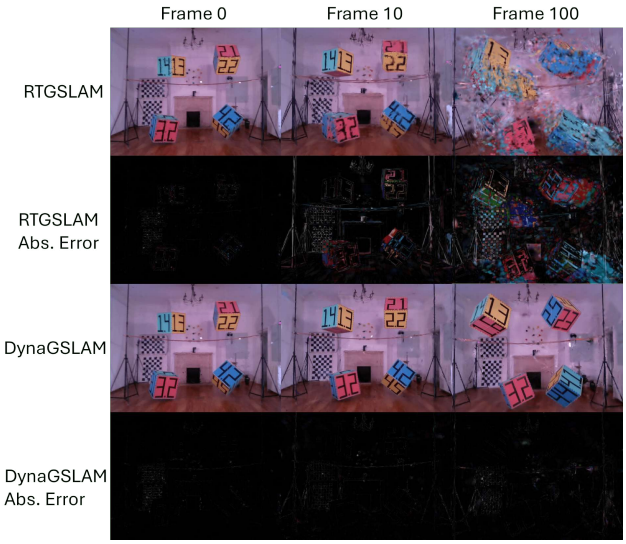

Figure 2. **Failures of static GS-SLAM over long frames.** "Avs. Error" is the absolute error of the rendered RGB to the ground truth RGB. Typical static GS-SLAM works (like RTGSLAM [54]) do not consider moving objects, not only the rendering quality of the moving objects becomes worse over long frames, but the static regions get contaminated by the failed dynamic GS. Our DynaGSLAM consistently renders both dynamic and static regions in high quality over long frames.

1. Introduction

Simultaneous localization and mapping (SLAM) is a fundamental problem with wide ranges of application in robotics [26], virtual reality, and autonomous vehicles. Given an input RGBD video sequence, a SLAM algorithm must jointly estimate the ego motion (camera pose) and build the map of the environment, conventionally represented as a point cloud. Although existing SLAM algorithms are robust and real-time in simple cases [47], there remain two major limitations that cannot be neglected: 1. Standard SLAM assumes a static scene, and cannot gracefully handle dynamic objects; 2. The pointcloud representation conveys basic 3D structure of the environment, but offers no photometric information on graphics or texture other than the input RGB, limiting downstream tasks.

Many efforts [6, 18, 21, 34, 35, 42, 46, 62, 79, 81] have been made to improve SLAM with moving objects. While implementations vary, the main idea is to detect and remove the moving objects that violate static assumptions for bundle adjustment. Some SOTA works [46, 79] take a step further to track object motions. This may be at the expense of computational speed; however, explicit consideration of moving objects improve robustness in many challenging indoor/outdoor scenes. Nonetheless, the pointcloud representation still offers limited photometric information.

With the rise of Gaussian splats (GS) [11, 24, 41, 48, 50, 51] as an explicit 3D representation with effective rendering speed and quality, recent works adopted GS to capture photometric information in SLAM [2, 17, 23, 44, 54, 67, 74, 77]. This body of work replaces the conventional pointclouds with GS, so that RGB images from novel views can be synthesized beyond the input camera views using GS. Such a representation retains the 3D structure with explicit GS, while adding significant photometric information. However, these SOTA methods only consider static scenes. Therefore, as shown in Fig. 2, even static regions are corrupted by the mishandling of dynamic regions. A few concurrent methods [27, 32, 33, 72, 73] propose "anti" dynamic SLAM with GS representation, where dynamic objects are detected and removed. Although this benefits localization, removing dynamic objects means that only the static background can be rendered from the GS map.

To represent dynamic objects with GS, SOTA methods [1, 12, 22, 25, 28, 30, 31, 38, 39, 43, 53, 59, 61, 66, 76, 78] explore directly adding a time dimension to GS. However, these methods train GS in an offline manner, for hours per video sequence, and are hence unsuitable for online SLAM. In contrast to all concurrent SOTA GS-SLAM or dynamic GS, our contributions can be summarized as:

- We propose a novel **dynamic GS management** algorithm for adding, deleting, tracking, updating and predicting dynamic GS. Our novel **online GS tracker** and the **motion interpolation/extrapolation** algorithms enable online real-time accurate dynamic GS mapping with reasonable memory requirements.
- By integrating our novel mapping module with existing real-time camera tracking system, we propose **DynaGSLAM**, the first real-time Gaussian-Splatting based SLAM that achieves high-quality online GS rendering, tracking, motion predictions of moving objects with dynamic ego motions in the scene.

2. Related Works

Dynamic SLAM. To handle dynamic objects, the usual approach is to detect and subtract the dynamic regions of the image. Earlier work [6, 62] use RANSAC [13] or point correlations [10] for motion detection, and the recent learning-based methods [3, 80] learn to semantically segment moving objects. These methods improve the quality of camera pose estimation by removing dynamic objects, but lose the object motion information. To extract the object motion, some dynamic SLAM incorporate and track the dynamic objects. These methods assume the objects are rigid, and assign a tracklet to every object. DynaSLAM2 [4] tracks rigid objects by estimating the motion of centroids, which also improves camera localization. SOTA work DynoSAM [46] proposes a world-centric factor-graph optimization for accuracy but suffers from time-consuming object motion estimation. Moreover, since all these attempts are based on classical point cloud mapping, it is non-trivial to directly

extend the ideas to GS-SLAM since GS have more complex attributes, to be optimized other than the point position, such as spherical harmonic and the shape (covariance).

GS-based SLAM. While Neural Radiance Field (NeRF) became popular for reconstruction [37, 45, 49] and used for SLAM [36, 57], it is too slow and lack of explicit representation. GS [24] has become a promising alternative to NeRF or point cloud for SLAM [23, 44, 54, 74, 77]. Initialized from an RGBD point cloud, new GS are incrementally added with ego motion to complete the scene. Compared to point cloud, 3DGS contains high-quality photometric information, at the expense of additional storage and computation. Thus, SOTA GS-SLAM focus on GS management algorithms, where RTGSLAM [54] designed a representative real-time algorithm for the tasks of adding, deleting, and reusing of static GS by converting GS in stable and unstable status. However, these SOTA GS-SLAM methods only work for static scenes.

There are concurrent efforts on extending GS-SLAM to dynamic scenes [27, 32, 33, 72, 73]. However, these methods fall under the "anti" dynamic SLAM category because they segment and discard dynamic objects. Our method, DynaGSLAM, is the first to construct a dynamic GS that models dynamic objects in the online SLAM setting.

Offline Dynamic GS. Dynamic GS [1, 12, 22, 25, 28, 30, 31, 38, 39, 43, 53, 59, 61, 66, 76, 78] has been also attracting lots of attention. With video and camera poses over frames, dynamic GS aims to train GS in "4D" such that the well-trained GS can be rendered from unseen views at any given timestamps in the video. [12, 76] explored an explicit additional time dimension for GS position and shape, and designed 4D rotation matrix with 4D-rotor and extend Spherical Harmonics to 4D, with the extra time dimension on all GS introduce speed and memory burden. The motion-function based dynamic GS [22, 28, 40] leveraged Fourier series & cubic polynomials for translation and SLERP (Spherical Linear Interpolation) for rotation, and embed the motion function parameters as GS attributes. However, these strategies introduce additional channels to all Gaussians, and require accurate supervision over time to learn the motion. Some methods can also segment moving GS [14, 30, 39, 78], but the motion-awareness is only achieved when the dynamic GS is fully trained over all frames. Furthermore, these methods all require long, offline training, and are not suitable for online SLAM. Compared to the offline dynamic GS methods, our dynamic GS-SLAM method performs equally well with three challenging constraints: 1) the target images are presented *online*, so future frames are inaccessible; 2) GS is optimized in real-time, while capturing dynamic objects; and 3) the camera trajectory is unknown or inaccurate.

3. Problem Formulation

We formulate the problem as follows: We are given a streaming sequence of RGB and depth images $C_t \in \mathbb{R}^{W\times H\times 3}$ and $D_t \in \mathbb{R}^{W\times H}$ of a scene, taken from known (4D mapping) or unknown (SLAM) camera poses $T_t \in SE(3)$. The scene contains moving objects.

The objective is to find a time-varying scene representation $\mathcal{G}_t$ that models both the static scene and the moving objects. We use the GS to represent $\mathcal{G}_t$, so that RGB and depth images $\hat{C}(\mathcal{G}_t, T_t)$ and $\hat{D}(\mathcal{G}_t, T_t)$ can be synthesized to match the scene at time t seen from camera pose T_t. With unknown camera pose, at each time t, we aim to find the camera trajectory $\{T_\tau\}$ for $\tau \in [0, t]$, and a time-varying scene representation $\{\mathcal{G}_\tau\}$, such that:

$$\min_{\mathcal{G}_\tau, T_\tau} \sum_{\tau=0}^{t} \ell_c(\hat{C}(\mathcal{G}_\tau, T_\tau), C_\tau) + \ell_d(\hat{D}(\mathcal{G}_\tau, T_\tau), D_\tau) \quad (1)$$

where ℓ_c and ℓ_d are color and depth image loss measuring the similarity between images $\hat{C}_t, \hat{D}_t$ reconstructed by GS and the images C_t, D_t provided by the camera. Furthermore, we focus on tracking a time-varying GS $\mathcal{G}_t$ over a time horizon, rather than only creating new GS for each timestep. This allows photorealistic synthesis of images not only at novel viewpoints (as in static GS), but also at *continuous* novel times. For simplicity, our notations reflect the case of regular, unit time intervals; however, we aim to predict and track motion over continuous time, given data arriving at irregular time intervals.

4. Dynamic GS Architecture

To solve the problem of photorealistic synthesis of dynamic objects, we introduce a new variant of GS with a dynamic mean that moves over time. We define a dynamic GS as a set of Gaussian blobs defined as $\mathcal{G}_t = \left\{(\mathbf{m}_t^i(\tau), \Sigma_t^i, \alpha_t^i, \mathbf{sh}_t^i)\right\}$, where $\Sigma_t^i \in \mathbb{R}^{3\times 3}$, $\alpha_t^i \in \mathbb{R}$ and $\mathbf{sh}_t^i \in \mathbb{R}^{16}$ are the covariance matrix, opacity and spherical harmonics. Importantly, $\mathbf{m}_t^i(\tau)$ is a time-varying mean allowing novel-view synthesis at an unobserved time τ, modeled as a cubic Hermite spline:

$$\begin{aligned}\mathbf{m}_t^i(\tau) = (2\tau'^3 - 3\tau'^2 + 1)\mathbf{m}_{t-}^i + (\tau'^3 - 2\tau'^2 + \tau')\mathbf{v}_{t-}^i \\ + (-2\tau'^3 + 3\tau'^2)\mathbf{m}_{t+}^i + (\tau'^3 - \tau'^2)\mathbf{v}_{t+}^i,\end{aligned} \quad (2)$$

where $\tau' = \tau - t - 1$, $\mathbf{m}_{t-,t+}^i$ and $\mathbf{v}_{t-,t+}^i$ are interpolation parameters. Notably, these parameters can be updated analytically without iterative optimization, as discussed later. Extrapolation into the future is achieved by querying $\tau > t$.

RGB images are rendered similarly to original 3DGS [24] through alpha-blending of projected Gaus-

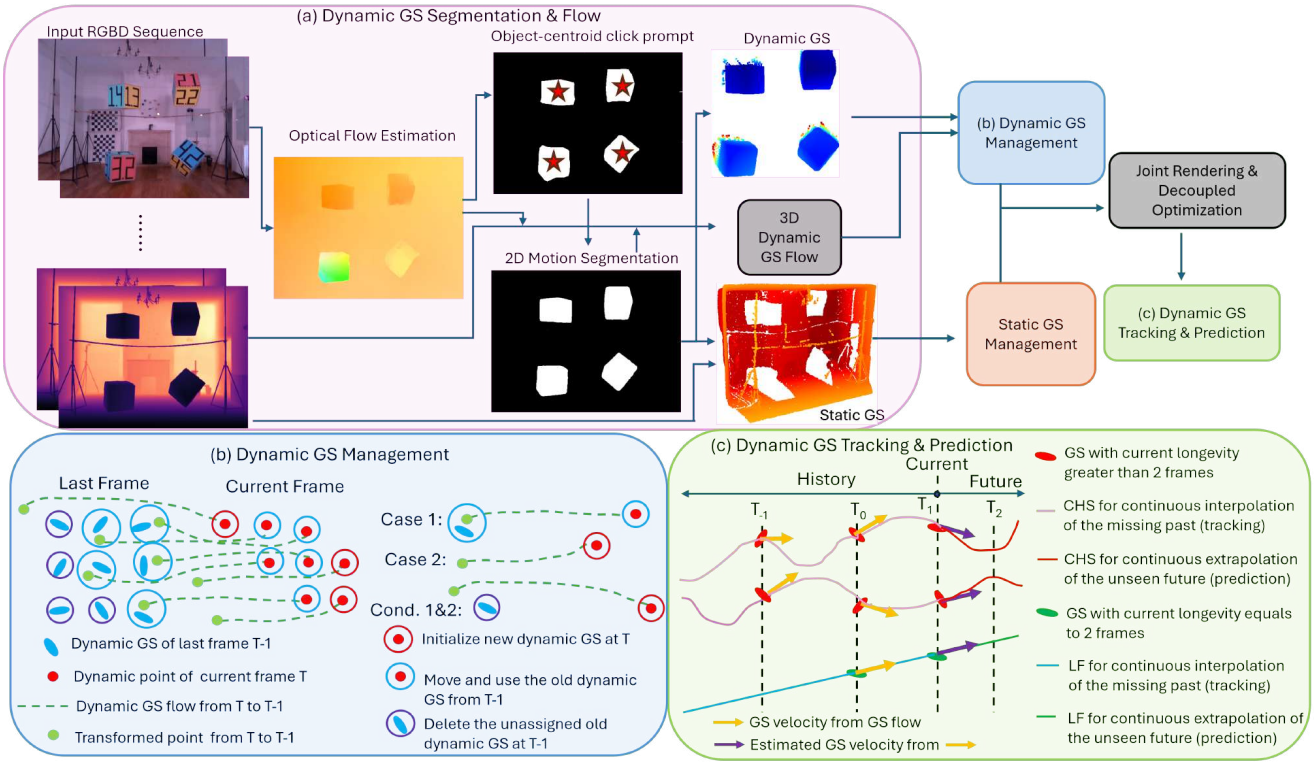

Figure 3. **Overview of DynaGSLAM Mapping.** We focus on three modules - Dynamic GS (a) Segmentation & Flow, (b) Management and (c) Tracking & Prediction. DynaGSLAM takes RGBD sequence as input to construct map with GS, (a) segment dynamic GS from static GS in 3D, and estimate dynamic GS 3D motion flow between frames. (b) Dynamic GS are managed separately from static GS with GS flow, but combined to jointly optimize. Case 1 &2 are the rules for dynamic GS adding; "Cond. 1&2" denotes the conditions for dynamic GS deletion. (c) The optimized dynamic GS at current and past frames are used to interpolate/extrapolate dynamic GS in the continuous timeline from past to future. "CHS" refers to "cubic Hermite spline" and "LF" refers to "linear function". Localization details are not included in the figure since our main contribution is on mapping.

sians at each time t:

$$\hat{C}_t = \sum_i c_t^i f(g_t^i) \prod_{j=1}^{i-1} (1 - f(g_t^j)), \tag{3}$$

where $f(g_t^i) = \alpha_t^i \mathcal{N}_{2D}(P\mathbf{m}_t^i(t), P\Sigma_t^i P^T)$ is the weight of the i-th Gaussian at time t, after affine projection P.

For rendering accurate depth images $\hat{D}_t$, we adopt the ideas from 2DGS [16], and discard the shortest principal axis of 3D Gaussians for higher efficiency and better surface representation. We also adopt the "surface rendering" technique for depth from [16] as it is faster than alpha blending. This takes the depth $d(g_t^i)$ of the closest GS that is over an opacity threshold λ_α:

$$\hat{D} = \min_{g_t^i \in \mathcal{G}_t} d(g_t^i), \text{ s.t. } f(g_t^i) \prod_{j=1}^{i-1} (1 - f(g_t^j)) > \lambda_\alpha. \tag{4}$$

5. Online Training of Dynamic GS

When training a static GS, the conventional approach is to first render RGB & depth images at target views as usual, and then minimize the loss between the rendered and observed ground truth, as is done in [23, 44, 54, 77]. However, this approach is insufficient for real-time training of dynamic GS because of the objects' motion. Here, we introduce an improved training method that first explicitly modifies the GS to match the current observations to accurately account for object motion.

5.1. Dynamic GS Flow

To make the most of the current observations, we utilize the optical flow at the current frame t to associate and propagate the existing GS from time $t-1$. To do so, we lift the 2D optical flow to 3D (Fig. 3(a)), and call it *dynamic GS flow*. To obtain the dynamic GS flow in 3D from the current to last frame $t \rightarrow t-1$, we mask out static optical flow with 2D motion mask M_t, project the moving optical flow $\mathbf{f}_{t-1\leftarrow t}(u,v)$ to 3D dynamic GS flow $\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t}$ using depth D_t, and compensate the ego motion:

$$\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t} = M_t \cdot \left(D_t K^{-1} \mathbf{f}_{t-1\leftarrow t}\right) - \left(T_{t-1\leftarrow t}\mathbf{P}_t - \mathbf{P}_t\right), \tag{5}$$

where K is the camera intrinsic, P is the camera pose, and $T_{t-1\leftarrow t}$ is the ego motion transformation.

5.2. Dynamic GS Management

Maintaining an appropriate number of GS is important for GS-SLAM, and even more so in the presence of dynamic objects. Adding new GS allows capturing the latest information, whereas adding too many will lead to prohibitively high memory usage (see the results of SplaTAM [23] on "rpy" in Table 2). Similarly, deleting old GS is essential for capping the memory usage, and more importantly to avoid introducing outliers in the future when it is outdated (see the results of RTGSLAM [54] in Fig. 2).

In this work, we present a management strategy for adding and deleting the dynamic GS. We store dynamic and static GS separately, with different addition and deletion strategies, although they are rendered jointly.

For static GS, we follow the strategy of SOTA static GS-SLAM [23, 54]. This strategy adds and optimizes new GS when new areas are seen, while keeping the old GS unchanged. If a GS is not seen for a certain number of frames, it is deleted.

For dynamic GS, this strategy is insufficient because of the objects' motion. We thus introduce a novel dynamic GS management algorithm shown in Fig. 3(b) that overcomes the limitations above. Let $\mathcal{G}_{t-1} = \{g^i_{t-1}\}$ be the GS up to time $t-1$, and let $\mathcal{P}_t = \{p^j_t\}$ be the current motion-segmented RGBD pointcloud. We first transform the current pointcloud $\mathcal{P}_t$ (red) to the previous timestep $t-1$ (blue), using the GS flow $\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t}$ (green) (5). Transforming the current pointcloud back in time using the latest optical flow observation $\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t}$ is better than using the cubic Hermite spline (2) for extrapolation, because the cubic Hermite spline only contains information up to time $t-1$. Moreover, by filtering the GS at time $t-1$, we avoid unnecessary forward-propagation of unnecessary GS to time t.

With the transformed pointcloud $\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t}\mathcal{P}_t$ (green), we search for the nearest neighbor in the existing dynamic GS (blue), and compare the nearest neighbor distance $d_{\min}(p^i_t)$ the average nearest neighbor distance $\bar{d}$, computed as:

$$\bar{d} = \frac{1}{N_W}\sum_{i=1}^{N_W} d_{\min}(p^i_t) = \frac{1}{N_W}\sum_{i=1}^{N_W} \min_{g\in\mathcal{G}_t} \|\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t} p^i_t - g\|. \tag{6}$$

We check if the nearest neighbor distance exceeds some ratio threshold λ_d of $\bar{d}$, resulting in two cases:

Case 1 (prev. observed points): $d_{\min}(p^i_t) \le \lambda_d \bar{d}$. The nearest past GS (blue) is within the distance threshold of the transformed point (green). In this case, we simply reuse the past GS (blue), by replacing their mean with the current point matched (red), and optimize using the current RGBD. This explicit modification is the key to higher performance, as it moves the past GS to the right location in one step, whereas the usual gradient updates as in the static GS case can only provide minor, insufficient displacements.

Case 2 (new points): $d_{\min}(p^i_t) > \lambda_d \bar{d}$. There is no past GS (blue) near the transformed point (green). In this case, a new GS (red) is initialized for the point p^i_t. This allows complete coverage of the whole scene, as some objects are unseen before the current frame (e.g. occluded sides of the moving box in Fig. 2).

We also check the validity of the existing GS against two conditions: observability and longevity.

Cond. 1 (observability): $\exists p^i_t \in \mathcal{P}_t,\ \|\mathbf{F}^{\text{Dyna}}_{t-1\leftarrow t} p^i_t - g_j\| \le \lambda_d \bar{d}$. We only keep GS (blue) that are within distance threshold of the currently observed points (green). This step is essential for dynamic GS because old, unobserved GS become outlier noise in the future if they are not displaced or deleted, unlike in static GS where the scene is unchanged.

Cond. 2 (longevity): We delete any GS that persisted for a longer period of time than a set longevity threshold.

As observed, the distance threshold ratio λ_d plays a crucial role in the dynamic GS management. We thoroughly study its impact in the ***Supplementary Sec D***. .

Point trackers may seem to serve the same purpose, but they do not, let alone being too slow for online SLAM ([29, Table 1]). We are searching for correspondences from the current pointcloud to the past GS, whereas point trackers predict the location of past points in the current frame, necessitating the management logic we presented.

5.3. Rendering and Optimization

Although being managed separately, we jointly render dynamic & static GS because it improves their interactions to better handle occlusions, lighting consistency, and spatial coherence. We follow the SOTA GS-SLAM [23, 44, 54, 74, 77] with similar supervision between the rendered ((3), (4)) and input RGBD over a small time window of past views.

Although jointly rendered, the optimization is decoupled in that different learning rates and longevity windows are used between dynamic and static GS. Static GS attributes remain more stable across frames, while dynamic GS undergo abrupt changes, decoupling ensures that their updates do not interfere with the optimization of static structures. Without decoupling, motion inconsistencies from dynamic objects introduce ghosting effects, or blending in the static region (as shown in Fig. 2), where remnants of dynamic objects appear in static regions due to incorrect optimization of photometric and geometric consistency.

Optimization-free update of motion spline. After associating and training the dynamic GS with respect to the current frame, the cubic Hermite spline (2) can be updated analytically without optimization. This is because the parameters $\mathbf{m}^i_{t-,t+}$ and $\mathbf{v}^i_{t-,t+}$ in (2) correspond exactly to the 3D position and the velocity of the center of GS g^i_t at the last $(t-)$ and current $(t+)$ frame. Thus, we directly set $\mathbf{m}^i_{t-,t+}$

from the optimized GS center at the last $(t-)$ and current $(t+)$ frames. The velocity term at the last frame $\mathbf{v}^i_{t-}$ is set as the negative of the GS flow (5), so that $\mathbf{v}^i_{t-} = -\mathbf{F}^{\text{Dyna}}_{(t-1)\leftarrow t}$. The velocity term at the current frame $\mathbf{v}^i_{t+}$ is estimated using the constant acceleration assumption, by extrapolating between $\mathbf{v}^i_{(t-1)+}$ and $\mathbf{v}^i_{t-}$. If $\mathbf{v}^i_{(t-1)+}$ is unavailable (e.g. when the GS g^i_t was just initialized), we fall back to the constant velocity assumption.

6. Experiments

Baselines: We compare the performance of our DynaGSLAM algorithm against four other SOTA GS-SLAM methods: RTGSLAM [54], SplatTAM [23], GSSLAM [44] and GSLAM [77]. Although there are other concurrent "anti" dynamic GS-SLAM methods [27, 32, 33, 72, 73] that remove dynamic objects, we could not reproduce these methods because their code is unavailable. We compare against the reported results where possible, and include the detailed experimental setup in the ***Supplementary***.

Datasets: Prior works on static GS-SLAM are evaluated on synthetic datasets with static scenes [9, 63]. In contrast, we evaluate our method on real datasets with dynamic scenes. We use OMD [20] and the dynamic scenes from the TUM [64] and the Bonn [52] datasets.

Metrics	Scene	[54]	[44]	[23]	[77]	[72]*	Ours
PSNR↑	*balloon*	13.7	19.3	18.8	24.5	24.0	**28.4**
	balloon2	12.9	17.8	16.4	23.1	22.9	**28.3**
	ps_track	13.2	14.9	15.6	24.7	24.6	**28.0**
	ps_track2	13.5	15.9	13.7	24.6	24.2	**27.4**
SSIM↑	*balloon*	38.4	73.0	73.5	85.8	77.5	**93.1**
	balloon2	32.0	67.5	60.6	83.2	71.5	**93.3**
	ps_track	36.1	59.2	54.9	86.4	78.7	**93.0**
	ps_track2	38.1	69.9	46.9	86.2	77.3	**91.6**
LPIPS↓	*balloon*	67.9	43.9	38.8	26.6	32.5	29.3
	balloon2	71.2	46.2	51.2	27.2	39.4	**26.6**
	ps_track	69.2	54.8	53.4	24.3	32.8	28.9
	ps_track2	66.5	45.4	56.2	24.3	32.0	31.6
DynaPSNR↑	*balloon*	18.8	14.6	15.2	19.8	-	**32.5**
	balloon2	16.6	14.1	14.6	20.0	-	**32.8**
	ps_track	18.1	8.8	10.3	21.6	-	**32.1**
	ps_track2	17.4	7.6	8.7	22.2	-	**32.6**

Table 1. **Comparison on Bonn Dataset.** *: Reported results from [72] without reproduction due to unavailability of code. DynaPSNR unavailable for [72], because dynamic objects are removed.

Dynamic Mapping Results. Tables 1 and 2 show the quantitative comparison of the GS mapping, our DynaGSLAM achieves superior results that outperforms other SOTA GS-SLAM on all dynamic sequences. The superior results in DynaPSNR illustrates the efficacy of our dynamic GS management algorithm. Fig. 1 and 4 show some qualitative comparisons, where our rendering quality is better than other works, especially around dynamic objects such as the two people (Fig. 1) and the balloon (Fig. 4). **Please confer the videos in Supplementary Material for full comparison.** Our rendering results exhibit some minor "floater" artifacts, because we use very few numbers of GS for efficiency compared to others (Table 7). In contrast, SplaTAM [23] runs out-of-memory(OOM) after 500 frames because it fails to delete outlier dynamic GS and release memory.

Metrics	Scene	[54]	[23]	[44]	[77]	Ours
PSNR↑	*fr3_wk_xyz*	14.3	13.8	13.5	22.0	**27.5**
	fr3_wk_static	13.9	15.5	15.9	20.2	**26.9**
	fr3_wk_rpy	15.2	OOM	13.7	25.0	**27.4**
	fr3_wk_hs	13.1	11.9	12.3	24.7	**27.2**
SSIM↑	*fr3_wk_xyz*	45.4	40.8	38.2	80.7	**95.7**
	fr3_wk_static	52.9	60.5	54.7	73.5	**96.1**
	fr3_wk_rpy	51.7	OOM	40.3	88.3	**94.7**
	fr3_wk_hs	34.3	33.4	37.0	87.9	**94.5**
LPIPS↓	*fr3_wk_xyz*	59.7	64.1	58.0	23.7	**16.0**
	fr3_wk_static	53.9	41.7	42.4	29.4	**14.0**
	fr3_wk_rpy	56.7	OOM	53.9	16.8	21.1
	fr3_wk_hs	69.9	67.3	66.8	16.3	20.0
DynaPSNR↑	*fr3_wk_xyz*	17.0	12.3	12.9	23.5	**31.5**
	fr3_wk_static	16.6	12.1	12.7	22.5	**30.3**
	fr3_wk_rpy	16.5	OOM	12.8	26.1	**30.1**
	fr3_wk_hs	14.9	12.1	12.7	26.2	**30.7**

Table 2. **Comparison on TUM Dataset.** Our method outperforms others on all metrics. Best results boldfaced. OOM indicates out of memory.

Dynamic Motion Tracking & Prediction. Fig. 5 shows a qualitative comparison of tracking and prediction with DynaGSLAM. Tracking is evaluated by interpolating and rendering an intermediate target timestamp (t_3) given two start and end frames (t_0 and t_5). For prediction, we extrapolate and render a future (t_{10}) timestamp given the same input. This is an extremely difficult task, as we are only given one out of every five frames, which is temporally sparse, to reconstruct the 3rd and 10th frames that are unseen. Since previous methods do not model dynamic objects' motion in GS, we take RTGSLAM [54] as baseline, assuming no moving entities at t_5 and render at the target viewpoint. The results in Fig. 5 shows that our method accurately predicts the moving objects (moving boxes and people), which overlap with the motion mask (transparent white).

We also conduct quantitative ablation study on the "Motion Horizon" for GS tracking and prediction on OMD and TUM datasets, as shown in Table 3. As the input frame interval or the motion horizon grows, the difficulty for motion estimation is increasing, and tracking (interpolation) always performs better than prediction (extrapolation). While our DynaGSLAM's performance edge is huge under small motion horizon, the advantage gets unclear while the motion horizon is growing. This is because the PSNR metric is very sensitive to even a small displacement - The exact same two patterns get low PSNR if overlapping with a minor displacement. With the motion horizon grows, displacement errors are zoomed out. However, we argue that for a long "Motion Horizon", visual results give fairer comparisons, such as the accurate fitting of the moving objects' contour (cubes, peo-

Figure 4. **Qualitative Results on Bonn Dataset.** First row: RGB rendering. Second row: Difference between the rendering and the ground truth. Our DynaGSLAM outperforms all baseline static GS-SLAM on the mapping quality, especially at the moving balloon and person.

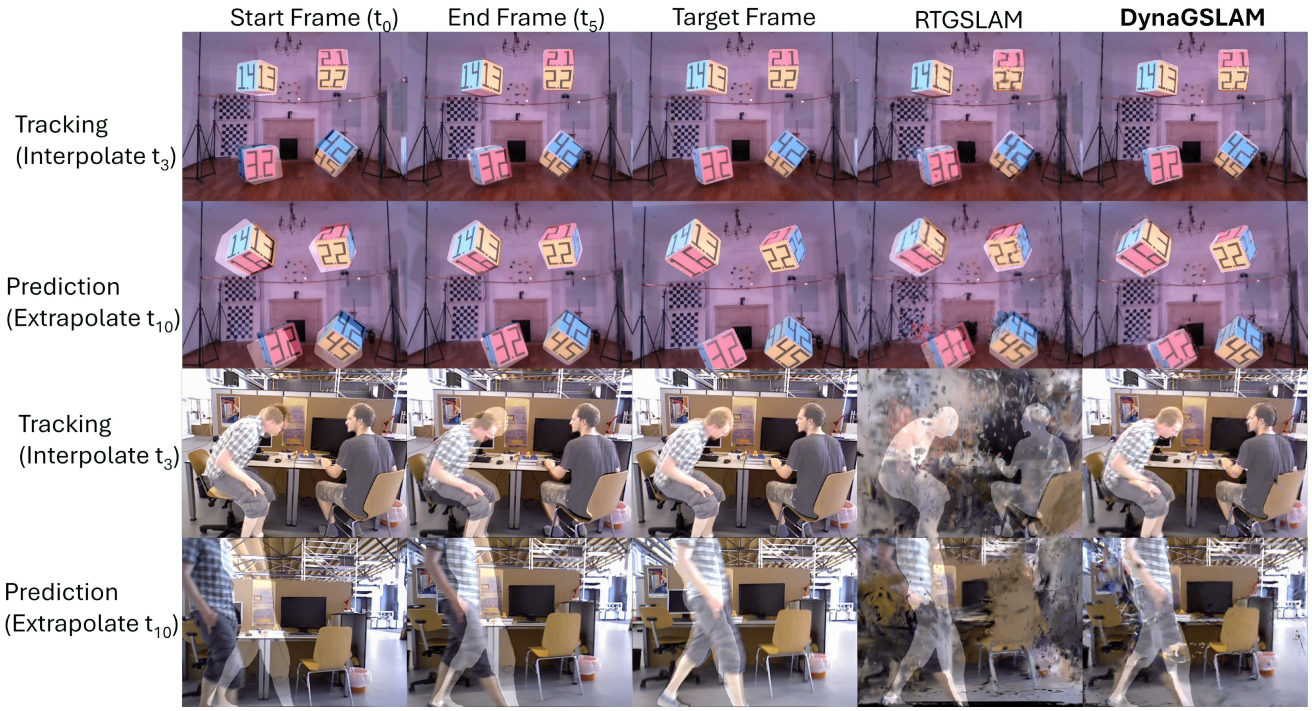

Figure 5. **Tracking & Prediction results on OMD and TUM dataset.** "Start/End Frame" denotes two consecutive frames (t_0 & t_5). "Target Frame" is the ground truth of t_3 (for tracking) and t_{10} (for prediction). Only "Start/End Frame" can be seen by SLAM, "Target Frame" cannot be seen by SLAM. When DynaGSLAM online proceeds to the "End Frame" (t_5) as the current frame, it interpolates the missing past frame t_3 and extrapolates the unseen future frame t_{10}. To better visualize motion quality, we overlap the ground-truth motion mask (white transparent) of the target frames to all frames of ground truths and estimations; A better overlapping between the moving objects and the masks indicates better motion estimation and rendering. Please zoom in to check details.

ple, balloons) with the ground truth motion mask in Fig. 5 and Fig. 12 (***Supplementary***).

Robustness to Segmentation Error We test the motion segmentation robustness with ***PWC-Net*** [65] and ***2-pixel Gaussian noise adding to our RAFT flow***, as shown in Fig. 6. Our dynamic masking algorithm can still detect robust centers as prompts for SAM2 and leads to similar performance under challenging optical flow maps, where the quantitative result is shown as PSNR/Dynamic PSNR in Table 4. We derived the algorithm based on the fact that moving objects have closed contour. With the edge detected, we performed Morphological closing (Dilation followed by Erosion) to ensure the motion region is enclosed, and then used flood-fill to fill the background. This learning-free al-

Method	OMD (S4U)		TUM (fr3_walking_static)	
	DynaGSLAM	RTGSLAM	DynaGSLAM	RTGSLAM
Mapping	30.63/34.58	17.12/15.66	26.88/30.30	13.92/16.59
Interval = 2 frames, Interpolate middle frame (1st)	20.69/16.69	16.90/15.19	20.83/19.92	14.25/17.07
Interval = 5 frames, Interpolate middle frame (3rd)	18.80/14.54	16.70/14.46	19.98/17.82	14.54/16.54
Interval = 2 frames, Extrapolate next 1st frame	20.45/16.10	16.51/14.55	18.64/15.51	14.09/16.58
Interval = 5 frames, Extrapolate next 1st frame	20.57/16.30	17.05/15.32	20.01/17.86	14.50/16.65
Interval = 5 frames, Extrapolate next 2nd frame	18.73/14.33	16.44/14.27	17.76/14.85	14.24/15.82
Interval = 5 frames, Extrapolate next 5th frame	17.24/13.30	15.65/13.14	15.82/14.84	13.99/12.27

Table 3. **Ablation Study on Motion Horizon for Tracking and Prediction.** In each cell the metric is represented as "PSNR↑/DynaPSNR↑". The conditions on "Interval = 5 frames, Interpolate middle frame (3rd)" and "Interval = 5 frames, Extrapolate next 5th frame" are the tracking and prediction corresponding to Fig. 5 and Fig. 12. Please check the annotation explanation in Fig. 5 to understand the condition in the table.

Original Setup (RAFT flow)	PWC-Net	RAFT flow with 2-px Gaussian Noise
30.6/34.6	30.8/34.5	30.3/33.3

Table 4. Robustness test of optical flow quality to the mapping performance as PSNR/DynaPSNR.

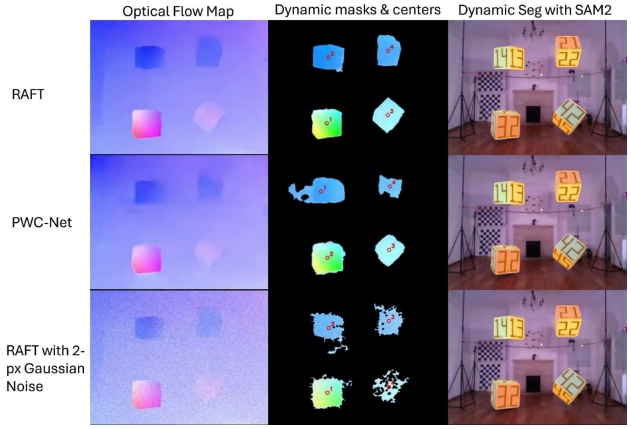

Figure 6. **2D preprocessing robustness**. Our real-time algorithm extract the centers of the dynamic objects as the mean of the dynamic masks, shown as red dots with numbers in the middle figures. Although the dynamic map is noisy, it is sufficient to provide the dynamic centers as the "click prompt" for online SAM2 segmentation, the masks are visualized in transparent orange. Please check the video at ***opt_seg.mp4 in Supp***.

gorithm runs fast and robust enough for real-time purposes. Ablation studies for the depth quality are in the ***Supp***. 2D motion segmentation is an important prior for our architecture, however, so far there is no perfect solution for online real-time motion segmentation. Although we made improvements on "automatic" segmentation, inaccurate masks can be generated due to three reasons: 1. SAM2 [55] loses moving objects in the tracking process, as shown in Fig. 7(abc), 2. The optical flow gradients are inaccurate for slow-moving objects, as shown in Fig. 7(d). 3. SAM2 tracker fails to perfectly segment fast-moving objects with

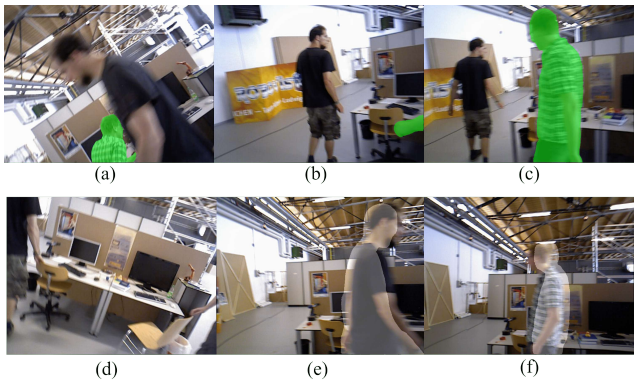

Figure 7. **Failure of 2D Motion Segmentation** further validates the robustness of our dynamic GS management algorithm under inaccurate motion priors. With some imperfect motion mask, our DynaGSLAM still enables to reasonably manage dynamic GS, and obtain outstanding mapping quality.

large motion blurs (as shown in 7(ef)). However, even with imperfect motion masks, DynaGSLAM still achieves outstanding GS mapping quality, which further validates the robustness of our dynamic GS management algorithm under inaccurate motion priors.

7. Conclusion and Future Works

We build the first online GS-based SLAM system - DynaGSLAM that render, track, and predict the motions of dynamic objects with ego motion estimation in real time. Our experiments on three real datasets validate the high quality, efficiency and robustness of our dynamic Gaussian mapping, along with accurate real-time ego motion estimation. To enable the online real-time usage, we pursue efficiency and sacrifice some complexity of the motion model. Future works should explore to various motion models/functions while keeping the system's efficiency.

8. Acknowledgment

This work was supported by XR Advanced Technology Group, Qualcomm Technologies, Inc, and by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the Strategic Technology Development Program supervised by the Korea Institute for Advancement of Technology (KIAT) [Grant No. P0026052].

References

[1] 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3

[2] Lizhi Bai, Chunqi Tian, Jun Yang, Siyu Zhang, Masanori Suganuma, and Takayuki Okatani. Rp-slam: Real-time photorealistic slam with efficient 3d gaussian splatting, 2024. 2

[3] Berta Bescos, JM. Fácil, Javier Civera, and José Neira. DynaSLAM: Tracking, mapping and inpainting in dynamic environments. *IEEE RA-L*, 2018. 2, 1

[4] Berta Bescos, Carlos Campos, Juan D. Tardós, and José Neira. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *IEEE Robotics and Automation Letters*, 6(3): 5191–5198, 2021. 2, 1

[5] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019. 1

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

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. 1

[8] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *ICCV*, 2023. 1

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 6

[10] Weichen Dai, Yu Zhang, Ping Li, Zheng Fang, and Sebastian Scherer. Rgb-d slam in dynamic environments using point correlations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[11] Bang Du, Runfa Blark Li, Chen Du, and Truong Nguyen. Glossgau: Efficient inverse rendering for glossy surface with anisotropic spherical gaussian, 2025. 2

[12] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. In *Proc. SIGGRAPH*, 2024. 2, 3

[13] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981. 2

[14] Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-aware 3d gaussian splatting for efficient dynamic scene reconstruction. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024. 3

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 1

[16] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. 4

[17] Huajian Huang, Longwei Li, Cheng Hui, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[18] Jiahui Huang, Sheng Yang, Zishuo Zhao, Yu-Kun Lai, and Shimin Hu. Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2, 1

[19] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. 2023. 1

[20] Kevin Michael Judd and Jonathan D. Gammell. The oxford multimotion dataset: Multiple se(3) motions with ground truth. *IEEE Robotics and Automation Letters*, 2019. 6

[21] Kevin M. Judd, Jonathan D. Gammell, and Paul Newman. Multimotion visual odometry (mvo): Simultaneous estimation of camera and third-party motions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. 2

[22] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. A compact dynamic 3d gaussian representation for real-time dynamic view synthesis. In *ECCV 2024*, 2024. 2, 3

[23] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, track & map 3d gaussians for dense rgb-d slam. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 3, 4, 5, 6, 1

[24] 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. 2, 3

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

[26] David Minkwan Kim, K. M. Brian Lee, Yong Hyeok Seo, Nikola Raicevic, Runfa Blark Li, Kehan Long, Chan Seon Yoon, Dong Min Kang, Byeong Jo Lim, Young Pyoung Kim, Nikolay Atanasov, Truong Nguyen, Se Woong Jun, and Young Wook Kim. A shared-autonomy construction robotic system for overhead works, 2025. 2

[27] Mangyu Kong, Jaewon Lee, Seongwon Lee, and Euntai Kim. Dgs-slam: Gaussian splatting slam in dynamic environment, 2024. 2, 3, 6

[28] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *ECCV*, 2024. 2, 3

[29] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Dense optical tracking: Connecting the dots. In *CVPR*, 2024. 5

[30] Junoh Lee, Chang-Yeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting, 2024. 2, 3

[31] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic gaussian fusion from casual videos via 4D motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024. 2, 3

[32] Haoang Li, Xiangqi Meng, Xingxing Zuo, Zhe Liu, Hesheng Wang, and Daniel Cremers. Pg-slam: Photo-realistic and geometry-aware rgb-d slam in dynamic environments, 2024. 2, 3, 6, 1

[33] Mingrui Li, Weijian Chen, Na Cheng, Jingyuan Xu, Dong Li, and Hongyu Wang. Garad-slam: 3d gaussian splatting for real-time anti dynamic slam, 2025. 2, 3, 6

[34] Runfa Li and Truong Nguyen. Sm3d: Simultaneous monocular mapping and 3d detection. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3652–3656, 2021. 2

[35] Runfa Li and Truong Nguyen. Monoplflownet: Permutohedral lattice flownet for real-scale 3d scene flow estimation with monocular images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 322–339. Springer Nature Switzerland, 2022. 2

[36] Runfa Li, Upal Mahbub, Vasudev Bhaskaran, and Truong Nguyen. Monoselfrecon: Purely self-supervised explicit generalizable 3d reconstruction of indoor scenes from monocular rgb views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 656–666, 2024. 3

[37] Runfa Blark Li, Keito Suzuki, Bang Du, Ki Myung Brian Lee, Nikolay Atanasov, and Truong Nguyen. Splatsdf: Boosting neural implicit sdf via gaussian splatting fusion, 2024. 3

[38] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3

[39] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. 2, 3

[40] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 3

[41] Xinshuang Liu, Runfa Blark Li, Keito Suzuki, and Truong Nguyen. Image-conditioned 3d gaussian splat quantization, 2025. 2

[42] Xinshuang Liu, Runfa Blark Li, Shaoxiu Wei, and Truong Nguyen. Importance-weighted non-iid sampling for flow matching models, 2025. 2

[43] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2, 3

[44] 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. 2, 3, 4, 5, 6

[45] 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. In *ECCV*, 2020. 3

[46] Jesse Morris, Yiduo Wang, Mikolaj Kliniewski, and Viorela Ila. Dynosam: Open-source smoothing and mapping framework for dynamic slam, 2025. 2, 5

[47] Raúl 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. 2

[48] Hung Nguyen, An Le, Blark Runfa Li, and Truong Nguyen. From coarse to fine: Learnable discrete wavelet transforms for efficient 3d gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3139–3148, 2025. 2

[49] Hung Nguyen, Blark Runfa Li, and Truong Nguyen. Dwt-nerf: Boosting few-shot neural radiance fields via discrete wavelet transform, 2025. 3

[50] Hung Nguyen, Runfa Li, An Le, and Truong Nguyen. Dwtgs: Rethinking frequency regularization for sparse-view 3d gaussian splatting, 2025. 2

[51] Hung Nguyen, Runfa Li, An Le, and Truong Nguyen. Waveletgaussian: Wavelet-domain diffusion for sparse-view 3d gaussian object reconstruction, 2025. 2

[52] 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

[53] Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. Splinegs: Robust motion-adaptive spline for real-time dynamic 3d gaussians from monocular video, 2024. 2, 3

[54] Zhexi Peng, Tianjia Shao, Liu Yong, Jingke Zhou, Yin Yang, Jingdong Wang, and Kun Zhou. Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting. 2024. 2, 3, 4, 5, 6

[55] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 8, 1, 5

[56] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1

[57] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022. 3

[58] Nicolas Schischka, Hannah Schieber, Mert Asim Karaoglu, Melih Gorgulu, Florian Grötzner, Alexander Ladikos, Nassir Navab, Daniel Roth, and Benjamin Busam. Dynamon: Motion-aware fast and robust camera localization for dynamic neural radiance fields. *IEEE Robotics and Automation Letters*, 10(1):548–555, 2025. 1

[59] Richard Shaw, Michal Nazarczuk, Jifei Song, Arthur Moreau, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Pérez-Pellitero. Swings: Sliding windows for dynamic 3d gaussian splatting. In *ECCV 2024*, 2024. 2, 3

[60] Shihao Shen, Yilin Cai, Wenshan Wang, and Sebastian Scherer. Dytanvo: Joint refinement of visual odometry and motion segmentation in dynamic environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4048–4055. IEEE, 2023. 1

[61] Nagabhushan Somraj, Kapil Choudhary, Sai Harsha Mupparaju, and Rajiv Soundararajan. Factorized motion fields for fast sparse input dynamic view synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, 2024. 2, 3

[62] Seungwon Song, Hyungtae Lim, Alex Junho Lee, and Hyun Myung. Dynavins: A visual-inertial slam for dynamic environments. *IEEE Robotics and Automation Letters*, 2022. 2

[63] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 6

[64] 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. 1, 6

[65] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 7, 1

[66] 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)*, 2024. 2, 3

[67] Lisong C. Sun, Neel P. Bhatt, Jonathan C. Liu, Zhiwen Fan, Zhangyang Wang, Todd E. Humphreys, and Ufuk Topcu. Mm3dgs slam: Multi-modal 3d gaussian splatting for slam using vision, depth, and inertial measurements, 2024. 2

[68] Keito Suzuki, Bang Du, Kunyao Chen, Runfa Li, and Truong Nguyen. Thp3d: Text-driven multi-granularity 3d human parsing. In *ECCV 2024 Workshops*, 2025. 5

[69] Keito Suzuki, Bang Du, Girish Krishnan, Kunyao Chen, Runfa Blark Li, and Truong Nguyen. Open-vocabulary semantic part segmentation of 3d human, 2025.

[70] Keito Suzuki, Bang Du, Runfa Blark Li, Kunyao Chen, Lei Wang, Peng Liu, Ning Bi, and Truong Nguyen. Openhuman4d: Open-vocabulary 4d human parsing, 2025. 5

[71] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV 2020*, 2020. 1, 5

[72] Long Wen, Shixin Li, Yu Zhang, Yuhong Huang, Jianjie Lin, Fengjunjie Pan, Zhenshan Bing, and Alois Knoll. Gassidy: Gaussian splatting slam in dynamic environments, 2024. 2, 3, 6, 1

[73] Yueming Xu, Haochen Jiang, Zhongyang Xiao, Jianfeng Feng, and Li Zhang. DG-SLAM: Robust dynamic gaussian splatting SLAM with hybrid pose optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3, 6, 1

[74] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024. 2, 3, 5

[75] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 1

[76] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. 2024. 2, 3

[77] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting, 2023. 2, 3, 4, 5, 6

[78] Daiwei Zhang, Gengyan Li, Jiajie Li, Mickaël Bressieux, Otmar Hilliges, Marc Pollefeys, Luc Van Gool, and Xi Wang. Egogaussian: Dynamic scene understanding from egocentric video with 3d gaussian splatting. *arXiv preprint arXiv:2406.19811*, 2024. 2, 3

[79] Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila. Vdo-slam: A visual dynamic object-aware slam system, 2021. 2, 1

[80] Tianwei Zhang, Huayan Zhang, Yang Li, Yoshihiko Nakamura, and Lei Zhang. Flowfusion: Dynamic dense rgb-d slam based on optical flow. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7322–7328, 2020. 2, 1

[81] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

DynaGSLAM: Real-Time Gaussian-Splatting SLAM for Online Rendering, Tracking, Motion Predictions of Moving Objects in Dynamic Scenes

Supplementary Material

A. Experimental Setup

For the Bonn dataset, we use the raw depth sensors measurements. For TUM and OMD datasets, we use DepthAnythingV2 [75] to get smooth depth and recover the real scale with the original depth map because the raw depth sensor measurements come with large portion of invalid regions. In addition to the common metrics (PSNR, SSIM, LPIPS) used for mapping evaluation, we evaluate "DynaPSNR" as PSNR only for dynamic objects within 2D motion masks. We evaluate the Absolute Trajectory Error (ATE) of camera localization. The experiments are conducted on a desktop with a single NVIDIA 3090Ti (24GB). For low-level image processing, we stick to pretrained models because 1. Jointly online per-frame optimizing the low-level 2D visual model is time and storage consuming, posing threats to real-time online purposes. 2. The disentanglement of the low-level 2D visual preprocessing model from 3D SLAM optimization enables convenient replacement of better 2D foundation models. Using pretrained semantic/optical/motion segmentation is a consensus as engineering-level image processing for SOTA dynamic SLAM even without GS rendering. SOTA and classical examples are shown as **"method {*pretrained 2D optical/semantic/motion segmentation models they use*}"**: DynaMoN [58] {*DeepLabV3* [7] + *DytanVO* [60]}; DGSLAM [73] {*OneFormer* [19] + *RAFT [71]*}; DGSSLAM [73] {*TrackAnything* [8]}; Gassidy [72] {*YOLO* [56]}; PG-SLAM [32] {*MaskRCNN*) [15]}; VDO-SLAM [79] {*MaskRCNN* [15] + *PWC-Net* [65]}; ClusterSLAM [18] {*MaskRCNN*}; DynaSLAM [3] {*MaskRCNN*}; DynaSLAM2 [4] {*YOLACT* [5]}; FlowFusion [80] {*PWC-Net*}. We use a combination of RAFT and a real-time online SAM2 [55] (Sec. B). Without our proposed novel online GS tracker and motion interpolation/extrapolation, simply segmenting motion masks cannot obtain high-quality dynamic GS rendering. Building a novel low-level 2D visual processing model is beyond the scope of this work.

B. Dynamic GS Segmentation & Flow

While online 3D motion segmentation is challenging and slow today, we propose a novel Dynamic GS segmentation strategy (Fig. 3). Our GS is initialized from the point cloud of RGBD, so we estimate 2D pixel motion and align them to the GS motion. When our SLAM proceeds to current frame t, given two consecutive images $C_t-1, C_t \in \mathbb{R}^{W\times H\times 3}$, we first use a real-time optical flow model (RAFT [71]) to estimate a flow image $f_{t-1\leftarrow t} \in \mathbb{R}^{W\times H\times 2}$, where each pixel stores its own 2D velocity, and the velocities of moving pixels are distinct from the static pixels. We then, compute the gradient of $f_{t-1\leftarrow t}$, which detect edges and close the shapes to get a coarse motion mask. By setting a click prompt at the object centroid, we use prompt-based model SAM2 [55] to segment 2D motion. Our strategy enables an automatic pipeline to segment moving pixels and filter out the static objects. Whereas the strategy handles well in general cases, it relies on robust optical flow.

Incorrect segmentation of static objects as dynamic does not deprecate GS quality since our dynamic GS management also handles static GS, it only introduces minor extra computation that could be ignored in practice. On the contrary, treating moving objects as static causes problems as classical static GS management cannot manage dynamic GS. However, our experiments show that our DynaGSLAM has tolerance to the low quality of 2D motion segmentations, which is further discussed in Sec 6 and Fig. 7.

	PSNR↑	SSIM↑	LPIPS↓	DynaPSNR↑
OMD (S4U)	30.6/31.0	95.1/95.7	15.1/15.1	34.6/31.1
TUM (xyz)	27.5/16.3	95.7/79.3	16.0/37.1	31.5/28.7
TUM (static)	26.9/12.8	96.1/77.7	14.0/37.8	30.3/27.0
TUM (rpy)	27.4/20.3	94.7/84.6	21.1/34.6	30.1/29.6
TUM (halfsphere)	27.2/19.4	94.5/83.1	20.0/35.4	30.7/31.6

Table 5. Ablation Study on the Impact of the Depth Quality. In each cell, the metric is "refined (DepthAnythingV2) depth/original sensor-depth".

C. Trick of Dynamic Mapping Results

A popular practice when evaluating mapping metrics (PSNR, SSIM, LPIPS) in previous works (like [23]) is to set all invalid pixels to be 0, where the invalid mask is defined by the invalid regions of the original depth maps. However, this practice is unfair since setting estimation and ground truth pixel values both to 0 significantly benefits all metrics. For a fair evaluation, we disregard the benefits from this practice by evaluating without mask, which is why our implementation results of SplaTAM [23] in Table 1 is worse than the results proposed in Table 3 of [72]. However, even without the boost of this practice, our mapping accuracy is significantly better than SOTA baselines.

D. Ablation Studies

Impact of depth quality. While our model is not very sensitive to noisy 2D motion priors, it relies on good depth. While the three datasets used in our work are all real

Dist Threshold λ_d	0	0.01	0.1	0.5
PSNR↑	30.63	29.15	26.15	23.95
DynaPSNR↑	34.58	28.20	20.21	17.04
SSIM↑	95.13	93.88	90.89	87.49
LPIPS↓	15.13	17.51	22.88	27.50
Reuse Rate (%)	0	22.75	56.36	83.53
Dyna GS Num ↓	48.0k	42.8k	38.3k	25.4k

Table 6. **Ablation Study** on the distance threshold of the nearest neighbor in Dynamic GS management. Reuse Rate is the ratio of dynamic GS that are used for at least two consecutive frames. With higher distance threshold, the GS reuse rate is higher, and less new GS are initialized, yielding lower mapping quality but higher efficiency with lower number of dynamic GS.

datasets with the depth from sensor, TUM's depth is unreliable. We achieved outstanding results on Bonn and OMD with their original depth (Table 1 and 5). However, the depth maps from TUM include large invalid regions, resulting point cloud in poor quality. Nonetheless, we use the original sensor-depth from OMD (Table 5) and Bonn (Table 1) to get outstanding mapping quality, which still proves the robustness of our DynaGSLAM with practical depth sensor resource. Moreover, although the PSNR with noisy depth is not ideal, its counterpart "DynaPSNR" is still competitive. The good mapping quality of the dynamic region regardless of the static scene further validates our proposed novel dynamic GS management.

Impact of the distance threshold. The distance threshold λ_d is the most important hyper-parameter for our dynamic GS management algorithm (Fig. 3(b) & Section 5.2). Table 6 shows the effect of distance threshold on mapping quality and computational efficiency. In our experiments, we chose a low distance threshold of $\lambda_d = 0.05$ with a limit of 50k dynamic GS for the best mapping quality. In other applications, it may be beneficial to trade off the mapping quality for computational efficiency.

E. Additional Results

Localization Results. Figure 8 visualizes the camera trajectory and ATE on OMD dataset (S4U). We directly adopt the world-centric graph optimization strategy from [46], which considers the moving objects. In contrast, ICP and ORBSLAM2 used in RTGSLAM do not distinguish between static and dynamic objects. The result validates the importance of static/dynamic separation.

Dynamic Mapping results. We show additional qualitative comparisons of the GS mapping quality between our DynaGSLAM with SOTA baseline GS-SLAM works (RTGSLAM[54], SplaTAM[23], GSSLAM [44], and GSLAM [77]). Fig. 10 is an extension of Fig. 4 on the Bonn Dataset. Fig. 11 is an extension of Fig. 1 on the TUM Dataset. Our DynaGSLAM significantly outperforms these baselines, especially around the moving object such as the

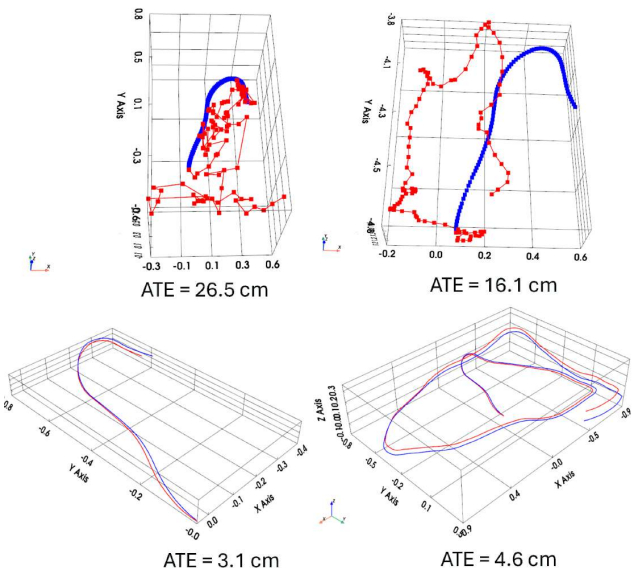

Figure 8. **Camera Tracking Results** on OMD dataset (S4U). Top Left: RTGSLAM [54]'s ICP, 90 frames. Top Right: RTGSLAM's ICP refined ORBSLAM2 [47], 90 frames. Bottom Left: Ours, 90 frames. Bottom Right: Ours, 500 frames.

Figure 9. **Mapping result on OMD dataset.**

balloon and moving people. The failures of the baseline GS-SLAM works can be attributed to two aspects: 1. The past dynamic GS cannot be effectively deleted with static GS management, which become outlier GS in the background and contaminate static GS, such as the remnant red GS noises of RTGSLAM in Fig. 10, which belong to the red hoodie of the person in the past frames. 2. The new GS cannot be effectively added with static GS management, such as the missing left leg of GSLAM in 11 (row 1). Our novel proposed dynamic GS management algorithm overcomes all these limitations proves to be robust and accurate in the real dataset. We also include one more full mapping result as complementary to Fig. 2. We attach one more frame shot from the video sequence as shown in Fig. 9.

Dynamic Motion Tracking & Prediction Results. We show additional qualitative results of tracking & prediction in Fig. 12. In all of the three datasets, our DynaGSLAM shows the ability to synthesize unseen views by traversing the time dimension. As an extension of Fig. 5, we show the tracking (interpolation) and challenging prediction (extrapolation) over many missing frames. With the transparent white mask as the ground truth motion, we show that our

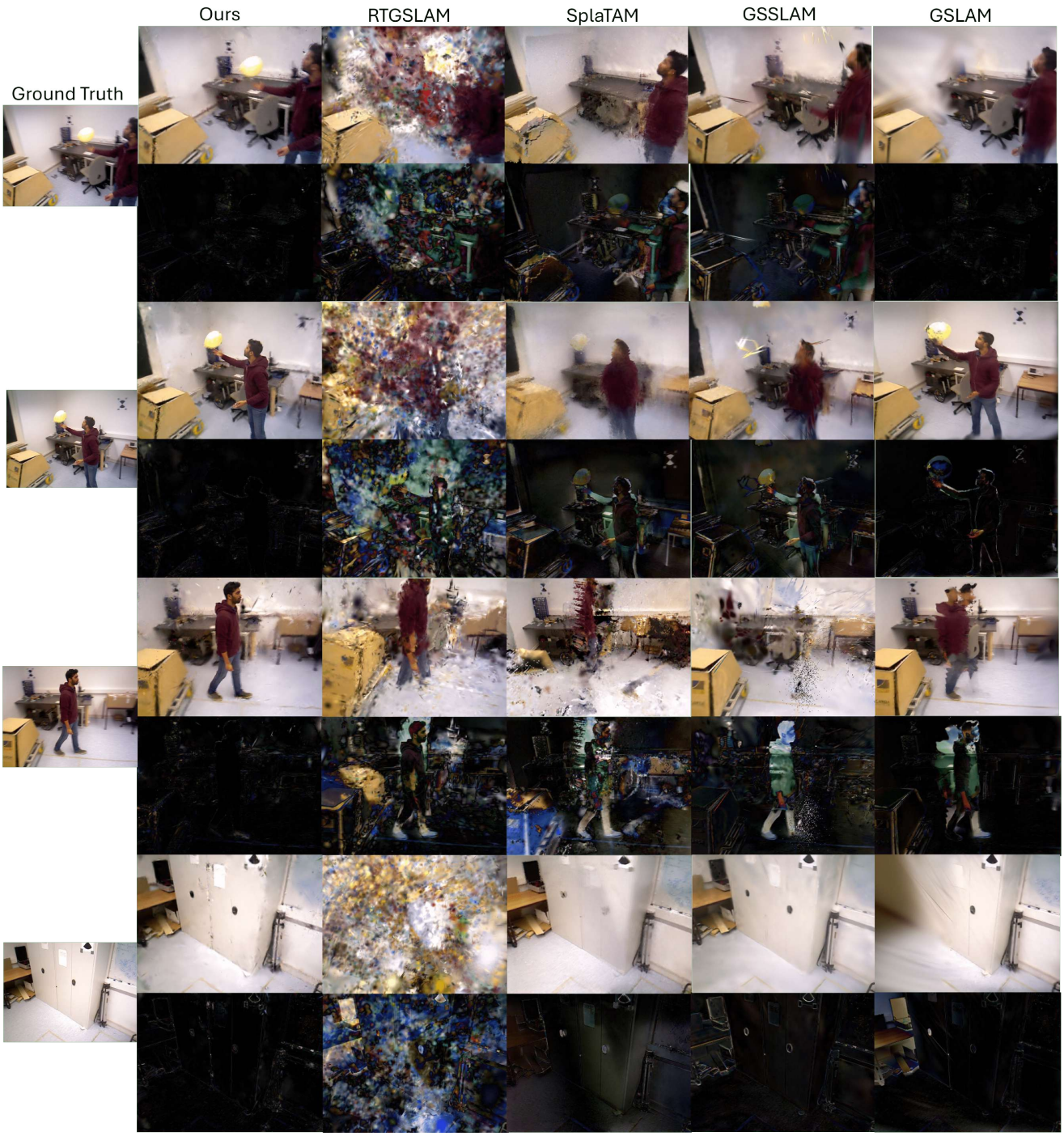

Figure 10. **GS Mapping Rendering Comparisons on the Bonn Dataset.** From top to bottom, the four scenes are: $balloon$, $balloon2$, ps_track, ps_track2. For each scene, the first row shows the RGB rendering results, the second row shows the absolute error between the rendered RGB to the ground truth. Our DynaGSLAM is obviously better than other SOTA GS-SLAM, especially at the moving entities such as the yellow balloon and the person.

motion model successfully brings GS to the desirable position, and the overlap of the dynamic entities (balloons and people) with the ground truth motion mask shows the quality of our proposed novel motion function. In contrast, the SOTA static GS-SLAM "RTGSLAM" [54] fails to correct the motion. Due to the lack of dynamic GS management,

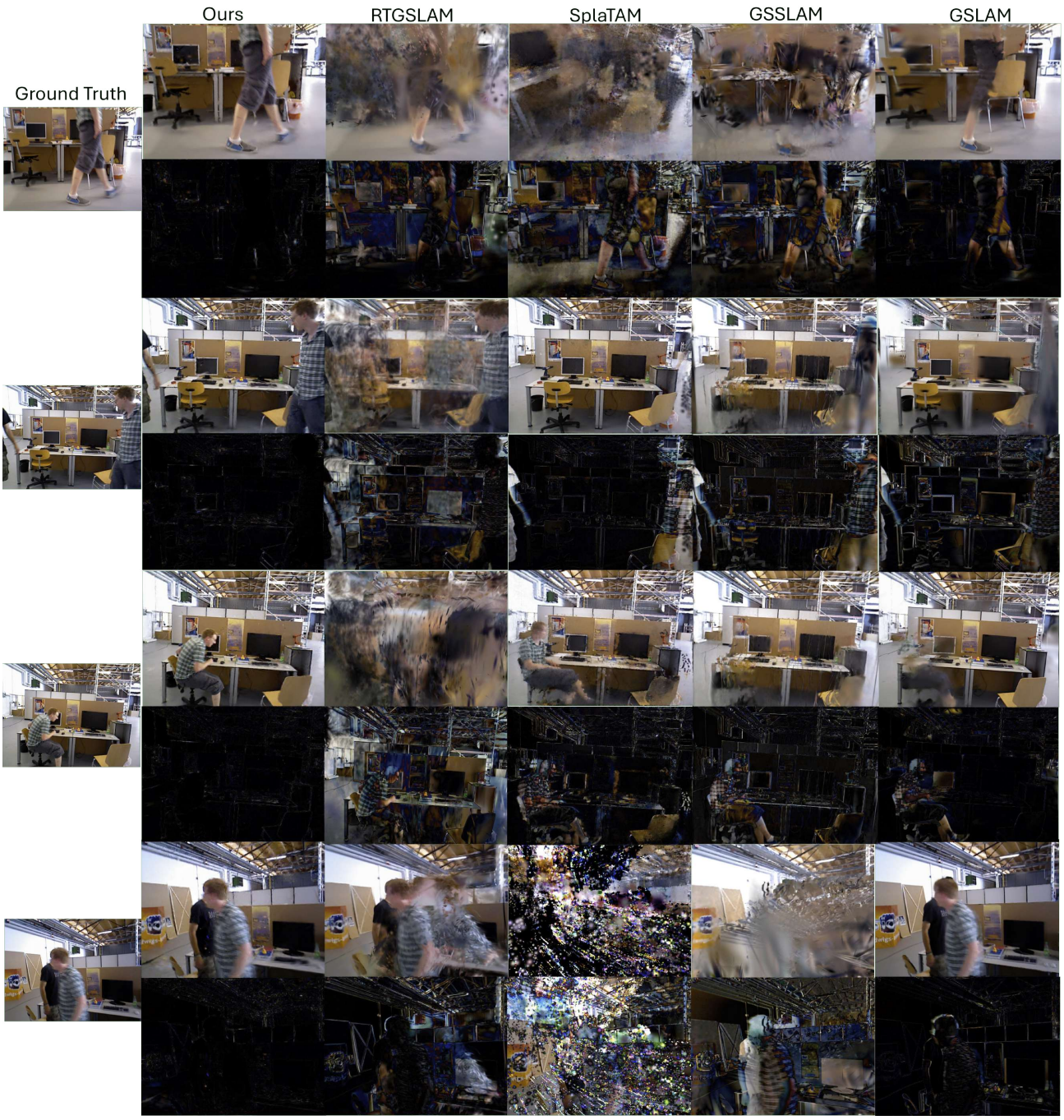

Figure 11. **GS Mapping Rendering Comparisons on the TUM Dataset.** From top to bottom, the four scenes are: $fr3\_walking\_xyz$, $fr3\_walking\_static$, $fr3\_walking\_static$, $fr3\_walking\_halfsphere$. For each scene, the first row shows the RGB rendering results, the second row shows the absolute error between the rendered RGB to the ground truth. Our DynaGSLAM is obviously better than other SOTA GS-SLAM, especially at the moving people.

their background static GS are also contaminated by moving GS. Our DynaGSLAM generates some minor artifacts under the extrapolation of long "Motion Horizon", which is mainly because we use an extremely low number of GS for real-time efficiency, so that individual GS can adjust the position and shape to cover more space whereas diminish their photometric textures, this issue can be moderated by trading-off the number and efficiency of GS.

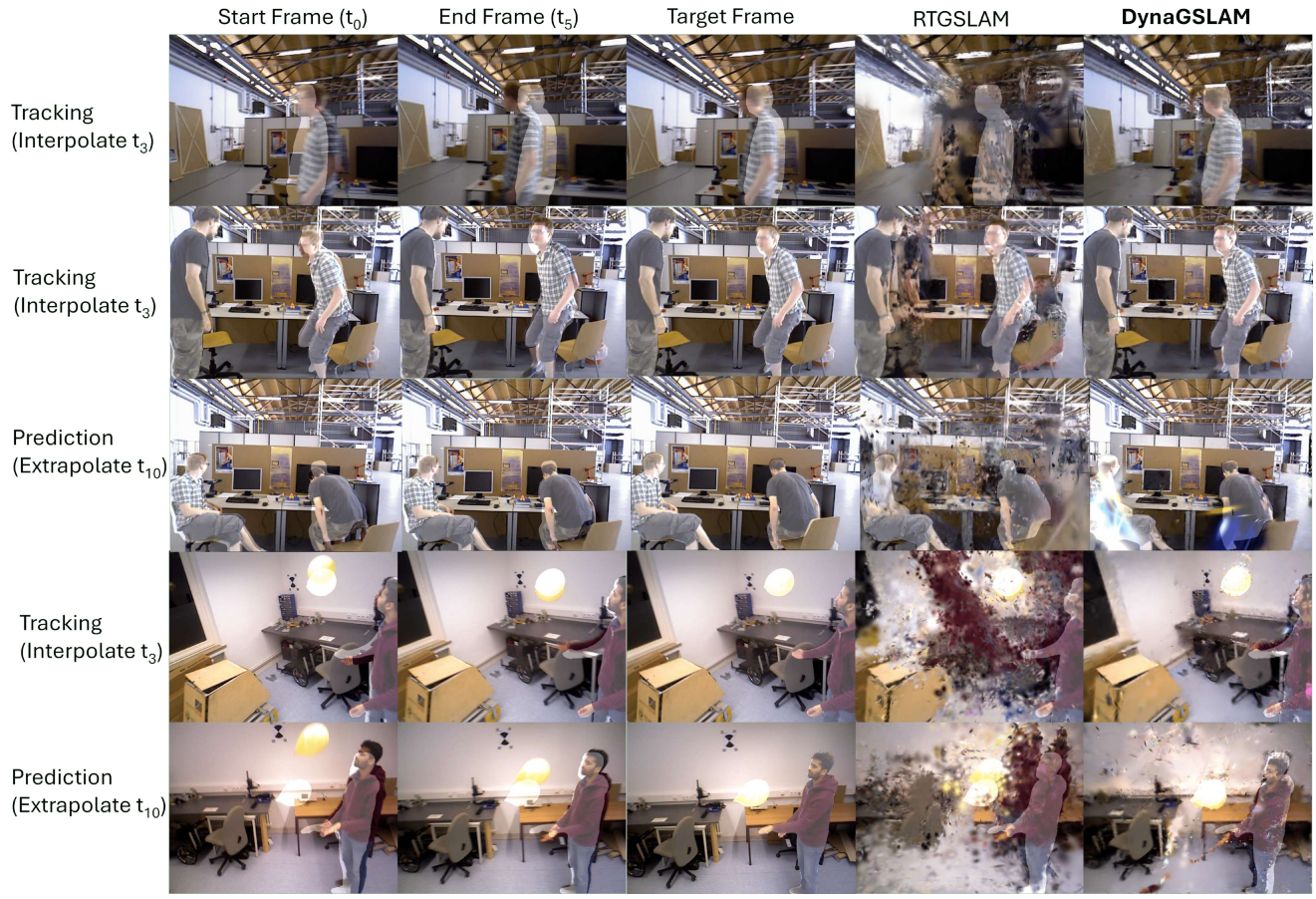

Figure 12. **Tracking & Prediction results on OMD, TUM and Bonn datasets.** This figure is an extension of Fig. 5 showing the tracking and prediction quality of our DynaGSLAM with our proposed novel GS motion function. Please check the annotation explanation in Fig. 5.

Methods	SplaTAM [23]	RTGSLAM [54]	Ours
Mapping (ms/frame)	1027	555	347
Localization (ms/frame)	5179	59	95
GS number	310K	520K	22K
Memory (GB)	0.82	1.7	2.6

Table 7. **Comparison of Inference Speed & Memory** on TUM fr3_walking_xyz with a single 3090Ti GPU.

Online Speed & Memory. Computational speed and memory usage are important for online SLAM. We compare online speed and memory usage on TUM "fr3_wk_xyz" sequence. The results are shown in Table 7. For SplaTAM [23], we follow their official configurations for TUM with 200 iters/frame for mapping optimization. For both RTGSLAM [54] and our DynaGSLAM, we follow RTGSLAM's official configuration with 50 iters/frame for mapping optimization. SplatTAM and RTGSLAM only update active GS to reduce the memory usage, but their static GS management fails on dynamic scenes yielding much larger number of GS than ours. Thanks to our dynamic GS management strategy, we achieve better mapping with much fewer GS. Our mapping runs at ∼347 ms/frame, including the online SAM2 [55] segmentation (∼36ms) and RAFT [71] optical flow (∼55ms). We chose SAM2 since its stability has been widely validated [68–70]. Our memory usage is higher than baselines mainly due to online SAM2 (∼1100mb) and RAFT (∼980mb), but it is acceptable given the benefits of dynamic rendering over baselines. We adapt DynoSAM for localization [46] (details in ***Supp***) that costs ∼450 ms/frame. Overall, the fast computation guarantees real-time operation with efficient memory usage, while ensuring accurate mapping and localization.